Journal of Artificial Intelligence Research 28 (2007) 299-348 Submitted 07/06; published 03/07

Supporting Temporal Reasoning by Mapping Calendar Expressions to Minimal Periodic Sets

Claudio Bettini bettini@dico.unimi.it
Sergio Mascetti mascetti@dico.unimi.it
Dipartimento di Informatica e Comunicazione, Università di Milano
Via Comelico, 39, 20135, Milan, Italy

X. Sean Wang Sean.Wang@uvm.edu
Department of Computer Science, University of Vermont
33 Colchester Avenue, Burlington, VT, 05405 USA

Abstract

In the recent years several research efforts have focused on the concept of time granularity and its applications. A first stream of research investigated the mathematical models behind the notion of granularity and the algorithms to manage temporal data based on those models. A second stream of research investigated symbolic formalisms providing a set of algebraic operators to define granularities in a compact and compositional way. However, only very limited manipulation algorithms have been proposed to operate directly on the algebraic representation making it unsuitable to use the symbolic formalisms in applications that need manipulation of granularities.

This paper aims at filling the gap between the results from these two streams of research, by providing an efficient conversion from the algebraic representation to the equivalent low-level representation based on the mathematical models. In addition, the conversion returns a minimal representation in terms of period length. Our results have a major practical impact: users can more easily define arbitrary granularities in terms of algebraic operators, and then access granularity reasoning and other services operating efficiently on the equivalent, minimal low-level representation. As an example, we illustrate the application to temporal constraint reasoning with multiple granularities.

From a technical point of view, we propose an hybrid algorithm that interleaves the conversion of calendar subexpressions into periodical sets with the minimization of the period length. The algorithm returns set-based granularity representations having minimal period length, which is the most relevant parameter for the performance of the considered reasoning services. Extensive experimental work supports the techniques used in the algorithm, and shows the efficiency and effectiveness of the algorithm.

1. Introduction

According to a 2006 research by Oxford University Press, the word *time* has been found to be the most common noun in the English language, considering diverse sources on the Internet including newspapers, journals, fictions and weblogs. What is somehow surprising is that among the 25 most common nouns we find time granularities like *day*, *week*, *month* and *year*. We are pretty sure that many other time granularities like *business day*, *quarter*, *semester*, etc. would be found to be quite frequently used in natural languages. However, the way computer applications deal with these concepts is still very naive and mostly hidden in program code and/or based on limited and sometimes imprecise calendar support.

Temporal representation and reasoning has been for a long time an AI research topic aimed at providing a formal framework for common sense reasoning, natural language understanding, planning, diagnosis and many other complex tasks involving time data management. Despite the many relevant contributions, time granularity representation and reasoning support has very often been ignored or over-simplified. In the very active area of temporal constraint satisfaction, most proposals implicitly assumed that adding support for granularity was a trivial extension. Only quite recently it was recognized that this is not the case and specific techniques were proposed (Bettini, Wang, & Jajodia, 2002a). Even the intuitively simple task of deciding whether a specific instant is part of a time granularity can be tricky when arbitrary user-defined granularities like e.g., *banking days*, or *academic semesters* are considered.

Granularities and periodic patterns in terms of granularities are playing a role even in emerging application areas like inter-organizational workflows and personal information management (PIM). For example, inter-organizational workflows need to model and monitor constraints like: *Event2 should occur no later than two business days after the occurrence of Event1.* In the context of PIM, current calendar applications, even on mobile devices, allow the user to specify quite involved periodical patterns for the recurrence of events. For example, it is possible to schedule an event every last Saturday of every two months. The complexity of the supported patterns has been increasing in the last years, and the current simple interfaces are showing their limits. They are essentially based on a combination of recurrences based on one or two granularities taken from a fixed set (days, weeks, months, and years). We foresee the possibility for significant extensions of these applications by specifying recurrences over user-defined granularities. For example, the user may define (or upload from a granularity library) the granularity corresponding to the academic semester of the school he is teaching at, and set the date of the finals as the last Monday of each semester. A bank may want to define its *banking days* granularity and some of the bank policies may then be formalized as recurrences in terms of that granularity. Automatically generated appointments from these policies may appear on the devices of bank employees involved in specific procedures. We also foresee the need to show a user preferred view of the calendar. With current standard applications the user has a choice between a business-day limited view and a complete view, but why not enabling a view based on the users's *consulting-days*, for example? A new perspective in the use of mobile devices may also result from considering the time span in which activities are supposed to be executed (expressed in arbitrary granularities), and having software agents on board to alert about constraints that may be violated, even based on contextual information like the user location or traffic conditions. This scenario highlights three main requirements: a) a sufficiently expressive formal model for time granularity, b) a convenient way to define new time granularities, and c) efficient reasoning tools over time granularities.

Consider a). In the last decade significant efforts have been made to provide formal models for the notion of time granularity and to devise algorithms to manage temporal data based on those models. In addition to *logical* approaches (Montanari, 1996; Combi, Franceschet, & Peron, 2004), a framework based on periodic-set representations has been extensively studied (Bettini, Wang, & Jajodia, 2000), and more recently an approach based on strings and automata was introduced (Wijsen, 2000; Bresolin, Montanari, & Puppis, 2004). We are mostly interested in the last two approaches because they support the effective computation of basic operations on time granularities. In both cases the representation of granularities can be considered as a *low-level* one, with a rather involved specification in terms of the instants of the time domain.

Consider requirement b) above. Users may have a hard time in defining granularities in formalisms based on low-level representations, and to interpret the output of operations. It is clearly unreasonable to ask users to specify granularities by linear equations or other mathematical formalisms that operate directly in terms of instants or of granules of a fixed time granularity. Hence, a second stream of research investigated more *high-level* symbolic formalisms providing a set of algebraic operators to define granularities in a compact and compositional way. The efforts on this task started even before the research on formal models for granularity (Leban, McDonald, & Forster, 1986; Niezette & Stevenne, 1992) and continued as a parallel stream of research (Bettini & Sibi, 2000; Ning, Wang, & Jajodia, 2002; Terenziani, 2003; Urgun, Dyreson, Snodgrass, Miller, Soo, Kline, & Jensen, 2007).

Finally, let us consider requirement c) above. Several inferencing operations have been defined on low-level representations, including equivalence, inclusion between granules in different granularities, and even complex inferencing services like constraint propagation (Bettini et al., 2002a). Even for simple operations no general method is available operating directly on the high level representation. Indeed, in some cases, the proposed methods cannot exploit the structure of the expression and require the enumeration of granules, which may be very inefficient. This is the case, for example, of the granule conversion methods presented by Ning e at. (2002). Moreover, we are not aware of any method to perform other operations, such as equivalence or intersection of sets of granules, directly in terms of the high level representation.

The major goal of this paper is to provide a unique framework to satisfy the requirements a), b), and c) identified above, by adding to the existing results a smart and efficient technique to convert granularity specifications from the high-level algebraic formalism to the low-level one, for which many more reasoning tools are available. In particular, in this paper we focus on the conversion from the high-level formalism called *Calendar Algebra* (Ning et al., 2002) to the low-level formalism based on periodical sets (Bettini et al., 2000, 2002a). Among the several proposals for the high-level (algebraic) specification of granularities, the choice of Calendar Algebra has two main motivations: first, it allows the user to express a large class of granularities; For a comparison of the expressiveness of Calendar Algebra with other formalisms see (Bettini et al., 2000). Second, it provides the richest set of algebraic operations that are designed to reflect the intuitive ways in which users define new granularities. A discussion on the actual usability of this tool and on how it could be enhanced by a graphical user interface can be found in Section 6.2. The choice of the low-level formalism based on periodic-sets also has two main motivations: first, an efficient implementation of all the basic operations already exists and has been extensively experimented (Bettini, Mascetti, & Pupillo, 2005); second, it is the only one currently supporting the complex operations on granularities needed for constraint satisfaction, as it will be illustrated in more detail in Section 6.1.

The technical contribution of this paper is a hybrid algorithm that interleaves the conversion of calendar subexpressions into periodical sets with a step for period minimization. A central phase of our conversion procedure is to derive, for each algebraic subexpression, the periodicity of the output set. This periodicity is used to build the periodical representation of the subexpression that can be recursively used as operand of other expressions. Given a calendar algebra expression, the algorithm returns set-based granularity representations having minimal period length. The period length is the most relevant parameter for the performance both of basic operations on granularities and of more specialized ones like the operations used by the constraint satisfaction service. Extensive experimental work reported in this paper validates the techniques used in the algorithm, by showing, among other things, that (1) even large calendar expressions can be efficiently converted, and (2) less precise conversion formulas may lead to unacceptable computation time. This latter property shows the importance of carefully and accurately designed conversion formulas. Indeed, conversion formulas may seem trivial if the length of periodicity is not a concern. In designing our conversion formulas, we made an effort to reduce the period length of the resulting granularity representation, and thus render the whole conversion process computationally efficient.

In the next section we define granularities; several interesting relationships among them are highlighted and the periodical set representation is formalized. In Section 3 we define Calendar Algebra and present its operations. In Section 4 we describe the conversion process: after the definition of the three steps necessary for the conversion, for each algebraic operation we present the formulas to perform each step. In Section 5 we discuss the period minimality issue, and we report experimental results based on a full implementation of the conversion algorithm and of its extension ensuring minimality. In Section 6 we further motivate our work by presenting a complete application scenario. Section 7 reports the related work, and Section 8 concludes the paper.

2. Formal Notions of Time Granularities

Time granularities include very common ones like hours, days, weeks, months and years, as well as the evolution and specialization of these granularities for specific contexts or applications. Trading days, banking days, and academic semesters are just few examples of specialization of granularities that have become quite common when describing policies and constraints.

2.1 Time Granularities

A comprehensive formal study of time granularities and their relationships can be found in (Bettini et al., 2000). In this paper, we only introduce notions that are essential to show our results. In particular, we report here the notion of *labeled granularity* which was proposed for the specification of a calendar algebra (Bettini et al., 2000; Ning et al., 2002); we will show later how any *labeled granularity* can be reduced to a more standard notion of granularity, like the one used by Bettini et al. (2002a).

Granularities are defined by grouping sets of instants into *granules*. For example, each granule of the granularity `day` specifies the set of instants included in a particular day. A label is used to refer to a particular granule. The whole set of time instants is called *time domain*, and for the purpose of this paper the domain can be an arbitrary infinite set with a total order relationship, $\leq$.

Definition 1 *A* labeled granularity *G is a pair $(\mathcal{L}_G, M)$, where $\mathcal{L}_G$ is a subset of the integers, and M is a mapping from $\mathcal{L}_G$ to the subsets of the time domain such that for each pair of integers i and j in $\mathcal{L}_G$ with $i < j$, if $M(i) \neq \emptyset$ and $M(j) \neq \emptyset$, then (1) each element in $M(i)$ is less than every element of $M(j)$, and (2) for each integer k in $\mathcal{L}_G$ with $i < k < j$, $M(k) \neq \emptyset$.*

The former condition guarantees the "monotonicity" of the granularity; the latter is used to introduce the bounds (see Section 2.2).

We call $\mathcal{L}_G$ the *label set* and for each $i \in \mathcal{L}_G$ we call $G(i)$ a *granule*; if $G(i) \neq \emptyset$ we call it a *non-empty granule*. When $\mathcal{L}_G$ is exactly the integers, the granularity is called "full-integer labeled". When $\mathcal{L}_G = \mathbb{Z}^+$ we have the same notion of granularity as used in several applications, e.g., (Bettini et al., 2002a). For example, following this labeling schema, if we assume to map `day`(1) to the subset of the time domain corresponding to January 1, 2001, `day`(32) would be mapped to February 1, 2001, `b-day`(6) to January 8, 2001 (the sixth business day), and `month`(15) to March 2002. The generalization to arbitrary label sets has been introduced mainly to facilitate conversion operations in the algebra, however our final goal is the conversion of a labeled granularity denoted by a calendar expression into a "positive-integer labeled" one denoted by a periodic formula.

2.2 Granularity Relationships

Some interesting relationships between granularities follows. The definitions are extended from the ones presented by Bettini et al. (2000) to cover the notion of labeled granularity.

Definition 2 *If G and H are labeled granularities, then G is said to* group into *H, denoted $G \trianglelefteq H$, if for each non-empty granule $H(j)$, there exists a (possibly infinite) set S of labels of G such that $H(j) = \bigcup_{i \in S} G(i)$.*

Intuitively, $G \trianglelefteq H$ means that each granule of H is a union of some granules of G. For example, `day` $\trianglelefteq$ `week` since a week is composed of 7 days and `day` $\trianglelefteq$ `b-day` since each business day is a day.

Definition 3 *If G and H are labeled granularities, then G is said to be* finer than *H, denoted $G \preceq H$, if for each granule $G(i)$, there exists a granule $H(j)$ such that $G(i) \subseteq H(j)$.*

For example `business-day` is finer than `day`, and also finer than `week`.

We also say that G *partitions* H if $G \trianglelefteq H$ and $G \preceq H$. Intuitively G partitions H if $G \trianglelefteq H$ and there are no granules of G other than those included in granules of H. For example, both `day` and `b-day` group into `b-week` (business week, i.e., the business day in a week), but `day` does not partition `b-week`, while `b-day` does.

Definition 4 *A labeled granularity G_1 is a* label-aligned subgranularity *of a labeled granularity G_2 if the label set $\mathcal{L}_{G_1}$ of G_1 is a subset of the label set $\mathcal{L}_{G_2}$ of G_2 and for each i in $\mathcal{L}_{G_1}$ such that $G_1(i) \neq \emptyset$, we have $G_1(i) = G_2(i)$.*

Intuitively, G_1 has a subset of the granules of G_2 and those granules have the same label in the two granularities.

Granularities are said to be *bounded* when $\mathcal{L}_G$ has a first or last element or when $G(i) = \emptyset$ for some $i \in \mathcal{L}_G$. We assume the existence of an unbounded bottom granularity, denoted by $\bot$ which is full-integer labeled and groups into every other granularity in the system.

There are time domains such that, given any set of granularities, it is always possible to find a bottom one; for example, it can be easily proved that this property holds for each time domain that has the same cardinality as the integers. On the other hand, the same property does not hold for other time domains (e.g. the reals). However, the assumption about the existence of the bottom granularity is still reasonable since we address problems in which granularities are defined starting from a bottom one. The definition of a *calendar* as a set of granularities that have the same bottom granularity (Bettini et al., 2000) captures this idea.

2.3 Granularity Conversions

When dealing with granularities, we often need to determine the granule (if any) of a granularity H that covers a given granule z of another granularity G. For example, we may wish to find the month (an interval of the absolute time) that includes a given week (another interval of the absolute time).

This transformation is obtained with the *up* operation. Formally, for each label $z \in \mathcal{L}_G$, $\lceil z \rceil_G^H$ is undefined if $\nexists z' \in \mathcal{L}_H$ s.t. $G(z) \subseteq H(z')$; otherwise, $\lceil z \rceil_G^H = z'$, where z' is the unique index value such that $G(z) \subseteq H(z')$. The uniqueness of z' is guaranteed by the monotonicity [1] of granularities. As an example, $\lceil z \rceil_{\texttt{second}}^{\texttt{month}}$ gives the month that includes the second z. Note that while $\lceil z \rceil_{\texttt{second}}^{\texttt{month}}$ is always defined, $\lceil z \rceil_{\texttt{week}}^{\texttt{month}}$ is undefined if week z falls between two months. Note that if $G \preceq H$, then the function $\lceil z \rceil_G^H$ is defined for each index value z. For example, since $\texttt{day} \preceq \texttt{week}$, $\lceil z \rceil_{\texttt{day}}^{\texttt{week}}$ is always defined, i.e., for each day we can find the week that contains it. The notation $\lceil z \rceil^H$ is used when the source granularity can be left implicit (e.g., when we are dealing with a fixed set of granularities having a distinguished bottom granularity).

Another direction of the above transformation is the *down* operation: Let G and H be granularities such that $G \trianglelefteq H$, and z an integer. Define $\lfloor z \rfloor_G^H$ as the set S of labels of granules of G such that $\bigcup_{j \in S} G(j) = H(z)$.[2] This function is useful for finding, e.g., all the days in a month.

2.4 The Periodical Granules Representation

A central issue in temporal reasoning is the possibility of finitely representing infinite granularities. The definition of granularity provided above is general and expressive but it may be impossible to provide a finite representation of some of the granularities. Even labels (i.e., a subset of the integers) do not necessarily have a finite representation.

A solution has been first proposed by Bettini et al. (2000). The idea is that most of the commonly used granularities present a periodical behavior; it means that there is a certain pattern that repeats periodically. This feature has been exploited to provide a method for

1. Condition (1) of Definition 1.
2. This definition is different from the one given by Bettini et al (2000) since it also considers non contiguous granules of G.

finitely describing granularities. The formal definition is based on the *periodically groups into* relationship.

Definition 5 *A labeled granularity G* groups periodically into *a labeled granularity H ($G \trianglelefteq H$) if $G \trianglelefteq H$ and there exist positive integers N and P such that*

(1) for each label i of H, $i + N$ is a label of H unless $i + N$ is greater than the greatest label of H, and

(2) for each label i of H, if $H(i) = \bigcup_{r=0}^{k} G(j_r)$ and $H(i + N)$ is a non-empty granule of H then $H(i + N) = \bigcup_{r=0}^{k} G(j_r + P)$, and

(3) if $H(s)$ is the first non-empty granule in H (if exists), then $H(s + N)$ is non-empty.

The *groups periodically into* relationship is a special case of the group into characterized by a periodic repetition of the "grouping pattern" of granules of G into granules of H. Its definition may appear complicated but it is actually quite simple. Since G groups into H, any granule $H(i)$ is the union of some granules of G; for instance assume it is the union of the granules $G(a_1), G(a_2), \ldots, G(a_k)$. Condition (1) ensures that the label $i + N$ exists (if it not greater than the greatest label of H) while condition (2) ensures that, if $H(i + N)$ is not empty, then it is the union of $G(a_1 + P), G(a_2 + P), \ldots, G(a_k + P)$. We assume that $\forall r = 0 \ldots k$, $(j_r + P) \in \mathcal{L}_G$; if not, the conditions are considered not satisfied. Condition (3) simply says that there is at least one of these repetitions.

We call each pair P and N in Definition 5, a *period length* and its associated *period label distance*. We also indicate with R the number of granules of H corresponding to each groups of P consecutive granules of $\perp$. More formally R is equal to the number of labels of H greater or equal than i and smaller than $i + N$ where i is an arbitrary label of H. Note that R is not affected by the value of i.

The period length and the period label distance are not unique; more precisely, we indicate with P_H^G the period length of H in terms of G and with N_H^G the period label distance of H in terms of G; the form P_H and N_H is used when $G = \perp$. Note that the period length is an integer value. For simplicity we also indicate with one period of a granularity H a set of R consecutive granules of H.

In general, the *periodically groups into* relationship guarantees that granularity H can be finitely described (in terms of granules of G).

Definition 6 *If $G \trianglelefteq H$, then H can be finitely described by providing: (i) a value for P and N; (ii) the set $\mathcal{L}^P$ of labels of H in one period of H; (iii) for each $a \in \mathcal{L}^P$, the finite set S_a of labels of G, such that $H(a) = \bigcup_{i \in S_a} G(i)$; (iv) the labels of first and last non-empty granules in H, if their values are not infinite.*

In this representation, the granules that have labels in $\mathcal{L}^P$ are the only ones that need to be explicitly represented; we call these granules the *explicit granules.*

If a granularity H can be represented as a periodic set of granules of a granularity G, then there exists an infinite number of pairs (P_H^G, N_H^G) for which the periodically groups into relation is satisfied. If the relation is satisfied for a pair (P, N), then it can be proved that it can also be satisfied for each pair $(\alpha P, \alpha N)$ with $\alpha \in \mathbb{N}^+$.

Definition 7 *A periodic representation of a granularity H in terms of G is called* minimal *if the period length P used in the representation has the smallest value among the period lengths appearing in all the pairs (P_H^G, N_H^G) for which H periodically groups into G.*

If H is fully characterized in terms of G, it is possible to derive the composition, in terms of G, of any granule of H. Indeed, if $\mathcal{L}^P$ is the set of labels of H with values in $\{b, \ldots, b + N_H^G - 1\}$, and we assume H to be unbounded, the description of an arbitrary granule $H(j)$ can be obtained by the following formula. Given $j' = [(j-1) \bmod N_H^G] + 1$ and

$$k = \begin{cases} \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor\right) \cdot N_H^G + j' & \text{if } \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor\right) \cdot N_H^G + j' \geq b \\ \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor + 1\right) \cdot N_H^G + j' & \text{otherwise} \end{cases}$$

we have

$$H(j) = \bigcup_{i \in S_k} G\left(P_H^G \cdot \left\lfloor \frac{j-1}{N_H^G} \right\rfloor + i - P_H^G \cdot \left\lfloor \frac{k-1}{N_H^G} \right\rfloor\right).$$

Example 1 *Figure 1 shows granularities* `day` *and* `week_parts` *i.e., the granularity that, for each week, contains a granule for the working days and a granule for the weekend. For the sake of simplicity, we denote* `day` *and* `week_parts` *with D and W respectively. Since $D \trianglelefteq W$, W is fully characterized in terms of D. Among different possible representations, in this example we decide to represent W in terms of D by $P_W^D = 7$, $N_W^D = 2$, $\mathcal{L}_W^P = \{3, 4\}$, $S_3 = \{8, 9, 10, 11, 12\}$ and $S_4 = \{13, 14\}$. The composition of each granule of W can then be easily computed; For example the composition of $W(6)$ is given by the formula presented above with $j' = 2$ and $k = 4$. Hence $W(6) = D(7 \cdot 2 + 13 - 7 \cdot 1) \cup D(7 \cdot 2 + 14 - 7 \cdot 1) = D(20) \cup D(21)$.*

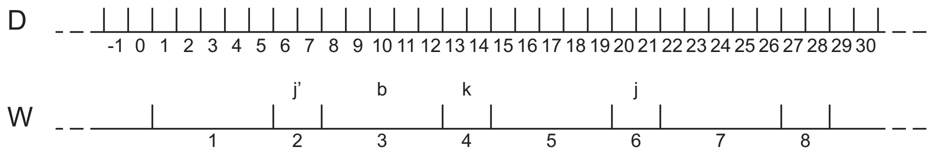


Figure 1: *Periodically groups into* example

3. Calendar Algebra

Several high-level symbolic formalisms have been proposed to represent granularities (Leban et al., 1986; Niezette & Stevenne, 1992).

In this work we consider the formalism proposed by Ning et al. (2002) called *Calendar Algebra*. In this approach a set of algebraic operations is defined; each operation generates a new granularity by manipulating other granularities that have already been generated. The relationships between the operands and the resulting granularities are thus encoded in the operations. All granularities that are generated directly or indirectly from the bottom granularity form a *calendar*, and these granularities are related to each other through the operations that define them. In practice, the choices for the bottom granularity include `day`, `hour`, `second`, `microsecond` and other granularities, depending on the accuracy required in each application context.

In the following we illustrate the calendar algebra operations presented by Ning et al. (2002) together with some restrictions introduced by Bettini et al. (2004).

3.1 The Grouping-Oriented Operations

The calendar algebra consists of the following two kinds of operations: the *grouping-oriented operations* and the *granule-oriented operations.* The grouping-oriented operations group certain granules of a granularity together to form new granules in a new granularity.

3.1.1 The Grouping Operation

Let G be a full-integer labeled granularity, and m a positive integer. The grouping operation $\mathit{Group}_m(G)$ generates a new granularity G' by partitioning the granules of G into m-granule groups and making each group a granule of the resulting granularity. More precisely, $G' = \mathit{Group}_m(G)$ is the granularity such that for each integer i,

$$G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i\cdot m} G(j).$$

For example, given granularity `day`, granularity `week` can be generated by the calendar algebra expression `week` $= \mathit{Group}_7($`day`$)$ if we assume that `day`(1) corresponds to Monday, i.e., the first day of a week.

3.1.2 The Altering-tick Operation

Let G_1, G_2 be full-integer labeled granularities, and l, k, m integers, where G_2 partitions G_1, and $1 \le l \le m$. The altering-tick operation $\mathit{Alter}^m_{l,k}(G_2, G_1)$ generates a new granularity by periodically expanding or shrinking granules of G_1 in terms of granules of G_2. Since G_2 partitions G_1, each granule of G_1 consists of some contiguous granules of G_2. The granules of G_1 can be partitioned into m-granule groups such that $G_1(1)$ to $G_1(m)$ are in one group, $G_1(m+1)$ to $G_1(2m)$ are in the following group, and so on. The goal of the altering-tick operation is to modify the granules of G_1 so that the l-th granule of every m-granule group will have $|k|$ additional (or fewer when $k < 0$) granules of G_2. For example, if G_1 represents 30-day groups (i.e., $G_1 = \mathit{Group}_{30}($`day`$)$) and we want to add a day to every 3-rd month (i.e., to make March to have 31 days), we may perform $\mathit{Alter}^{12}_{3,1}($`day`$, G_1)$.

The altering-tick operation can be formally described as follows. For each integer i such that $G_1(i) \neq \emptyset$, let b_i and t_i be the integers such that $G_1(i) = \cup_{j=b_i}^{t_i} G_2(j)$ (the integers b_i and t_i exist because G_2 partitions G_1). Then $G' = \mathit{Alter}^m_{l,k}(G_2, G_1)$ is the granularity such that for each integer i, let $G'(i) = \emptyset$ if $G_1(i) = \emptyset$, and otherwise let

$$G'(i) = \bigcup_{j=b'_i}^{t'_i} G_2(j),$$

where

$$b'_i = \begin{cases} b_i + (h-1)\cdot k, & \text{if } i = (h-1)\cdot m + l, \\ b_i + h\cdot k, & \text{otherwise,} \end{cases}$$

$$t'_i = t_i + h\cdot k,$$

and

$$h = \left\lfloor \frac{i-l}{m} \right\rfloor + 1.$$

Example 2 *Figure 2 shows an example of the* `Alter` *operation. Granularity G_1 is defined by $G_1 = Group_5(G_2)$ and granularity G' is defined by $G' = Alter^2_{2,-1}(G_2, G_1)$, which means shrinking the second one of every two granules of G_1 by one granule of G_2.*

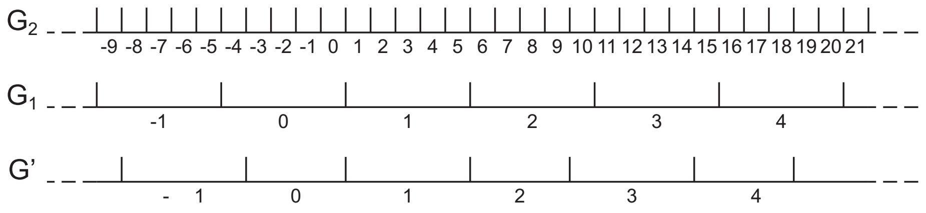


Figure 2: *Altering-tick* operation example

The original definition of altering-tick given by Ning et al. (2002) as reported above, has the following problems when an arbitrary negative value for k is used: (1) It allows the definition of a G' that is not a full-integer labeled granularity and (2) It allows the definition of a G' that does not even satisfy the definition of granularity. In order to avoid this undesired behavior, we impose the following restriction:

$$k > -(mindist(G1, 2, G2) - 1)$$

where $mindist()$ is formally defined by Bettini et al. (2000).

Intuitively, $mindist(G1, 2, G2)$ represents the minimum distance (in terms of granules of $G2$) between two consecutive granules of $G1$.

3.1.3 The Shift Operation

Let G be a full-integer labeled granularity, and m an integer. The shifting operation $Shift_m(G)$ generates a new granularity G' by shifting the labels of G by m positions. More formally, $G' = Shift_m(G)$ is the granularity such that for each integer i, $G'(i) = G(i-m)$. Note that G' is also full-integer labeled.

3.1.4 The Combining Operation

Let G_1 and G_2 be granularities with label sets $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$ respectively. The combining operation $Combine(G_1, G_2)$ generates a new granularity G' by combining all the granules of G_2 that are included in one granule of G_1 into one granule of G'. More formally, for each $i \in \mathcal{L}_1$, let $s(i) = \emptyset$ if $G_1(i) = \emptyset$, and otherwise let $s(i) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$.

Then $G' = \mathit{Combine}(G_1, G_2)$ is the granularity with the label set $\mathcal{L}_{G'} = \{i \in \mathcal{L}_{G_1} | s(i) \neq \emptyset\}$ such that for each i in $\mathcal{L}_{G'}$, $G'(i) = \bigcup_{j \in s(i)} G_2(j)$.

As an example, given granularities $\texttt{b-day}$ and $\texttt{month}$, the granularity for business months can be generated by $\texttt{b-month} = \mathit{Combine}(\texttt{month}, \texttt{b-day})$.

3.1.5 The Anchored Grouping Operation

Let G_1 and G_2 be granularities with label sets $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$ respectively, where G_2 is a label-aligned subgranularity of G_1, and G_1 is a full-integer labeled granularity. The anchored grouping operation $\mathit{Anchored\text{-}group}(G_1, G_2)$ generates a new granularity G' by combining all the granules of G_1 that are between two granules of G_2 into one granule of G'. More formally, $G' = \mathit{Anchored\text{-}group}(G_1, G_2)$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_2}$ such that for each $i \in \mathcal{L}_{G'}$, $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ where i' is the next label of G_2 after i.

For example, each academic year at a certain university begins on the last Monday in August, and ends on the day before the beginning of the next academic year. Then, the granularity corresponding to the academic years can be generated by $AcademicYear = \mathit{Anchored\text{-}group}(\texttt{day}, \texttt{lastMondayOfAugust})$.

3.2 The Granule-Oriented Operations

Differently from the grouping-oriented operations, the granule-oriented operations do not modify the granules of a granularity, but rather enable the selection of the granules that should remain in the new granularity.

3.2.1 The Subset Operation

Let G be a granularity with label set $\mathcal{L}_G$, and m, n integers such that $m \leq n$. The subset operation $G' = \mathit{Subset}_m^n(G)$ generates a new granularity G' by taking all the granules of G whose labels are between m and n. More formally, $G' = \mathit{Subset}_m^n(G)$ is the granularity with the label set $\mathcal{L}_{G'} = \{i \in \mathcal{L}_G \mid m \leq i \leq n\}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G(i)$. For example, given granularity $\texttt{year}$, all the years in the 20th century can be generated by $\texttt{20CenturyYear} = \mathit{Subset}_{1900}^{1999}(\texttt{year})$. Note that G' is a label-aligned subgranularity of G, and G' is not a full-integer labeled granularity even if G is. We also allow the extensions of setting $m = -\infty$ or $n = \infty$ with semantics properly extended.

3.2.2 The Selecting Operations

The selecting operations are all binary operations. They generate new granularities by selecting granules from the first operand in terms of their relationship with the granules of the second operand. The result is always a label-aligned subgranularity of the first operand granularity.

There are three selecting operations: *select-down*, *select-up* and *select-by-intersect*. To facilitate the description of these operations, the $\Delta_k^l(S)$ notation is used. Intuitively, if S is a set of integers, $\Delta_k^l(S)$ selects l elements starting from the k-th one (for a formal description of the Δ operator see (Ning et al., 2002)).

Select-down operation. For each granule $G_2(i)$, there exits a set of granules of G_1 that is contained in $G_2(i)$. The operation $\mathit{Select\text{-}down}_k^l(G_1, G_2)$, where $k \neq 0$ and $l > 0$ are integers, selects granules of G_1 by using $\Delta_k^l(\cdot)$ on each set of granules (actually their labels) of G_1 that are contained in one granule of G_2. More formally, $G' = \textit{Select-down}_k^l(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \cup_{i \in \mathcal{L}_{G_2}} \Delta_k^l(\{j \in \mathcal{L}_{G_1} \mid \emptyset \neq G_1(j) \subseteq G_2(i)\}),$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, Thanksgiving days are the fourth Thursdays of all Novembers; if `Thursday` and `November` are given, it can be generated by $\texttt{Thanksgiving} = \textit{Select-down}_4^1(\texttt{Thursday}, \texttt{November})$.

Select-up operation. The select-up operation $\textit{Select-up}(G_1, G_2)$ generates a new granularity G' by selecting the granules of G_1 that contain one or more granules of G_2. More formally, $G' = \textit{Select-up}(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \{i \in \mathcal{L}_{G_1} | \exists j \in \mathcal{L}_{G_2}(\emptyset \neq G_2(j) \subseteq G_1(i)), \}$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, given granularities `Thanksgiving` and `week`, the weeks that contain Thanksgiving days can be defined by $\texttt{ThanxWeek} = \textit{Select-up}(\texttt{week}, \texttt{Thanksgiving})$.

Select-by-intersect operation. For each granule $G_2(i)$, there may exist a set of granules of G_1, each intersecting $G_2(i)$. The $\textit{Select-by-intersect}_k^l(G_1, G_2)$ operation, where $k \neq 0$ and $l > 0$ are integers, selects granules of G_1 by applying $\Delta_k^l(\cdot)$ operator to all such sets, generating a new granularity G'. More formally, $G' = \textit{Select-by-intersect}_k^l(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \cup_{i \in \mathcal{L}_{G_2}} \Delta_k^l(\{j \in \mathcal{L}_{G_1} \mid G_1(j) \cap G_2(i) \neq \emptyset\}),$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, given granularities `week` and `month`, the granularity consisting of the first week of each month (among all the weeks intersecting the month) can be generated by $\texttt{FirstWeekOfMonth} = \textit{Select-by-intersect}_1^1(\texttt{week}, \texttt{month})$.

3.2.3 The Set Operations

In order to have the set operations as a part of the calendar algebra and to make certain computations easier, we restrict the operand granularities participating in the set operations so that the result of the operation is always a valid granularity: the set operations can be defined on G_1 and G_2 only if there exists a granularity H such that G_1 and G_2 are both label-aligned subgranularities of H. In the following, we describe the union, intersection, and difference operations of G_1 and G_2, assuming that they satisfy the requirement.

Union. The union operation $G_1 \cup G_2$ generates a new granularity G' by collecting all the granules from both G_1 and G_2. More formally, $G' = G_1 \cup G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cup \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$,

$$G'(i) = \begin{cases} G_1(i), & i \in \mathcal{L}_1, \\ G_2(i), & i \in \mathcal{L}_2 - \mathcal{L}_1. \end{cases}$$

For example, given granularities `Sunday` and `Saturday`, the granularity of the weekend days can be generated by $\texttt{WeekendDay} = \texttt{Sunday} \cup \texttt{Saturday}$.

Intersection. The intersection operation $G_1 \cap G_2$ generates a new granularity G' by taking the common granules from both G_1 and G_2. More formally, $G' = G_1 \cap G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cap \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$ (or equivalently $G_2(i)$).

Difference. The difference operation $G_1 \setminus G_2$ generates a new granularity G' by excluding the granules of G_2 from those of G_1. More formally, $G' = G_1 \setminus G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \setminus \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$.

4. From Calendar Algebra to Periodical Set

In this section we first describe the overall conversion process and then we report the formulas specific for the conversion of each calendar algebra operation. Finally, we present a procedure for relabeling the resulting granularity, a sketch complexity analysis and some considerations about the period length minimality.

4.1 The Conversion Process

Our final goal is to provide a correct and effective way to convert calendar expressions into periodical representations. Under appropriate limitations, for each calendar algebra operation, if the periodical descriptions of the operand granularities are known, it is possible to compute the periodical characterization of the resulting granularity.

This result allows us to calculate, for any calendar, the periodical description of each granularity in terms of the bottom granularity. In fact, by definition, the bottom granularity is fully characterized; hence it is possible to compute the periodical representation of all the granularities that are obtained from operations applied to the bottom granularity. Recursively, the periodical description of all the granularities can be obtained.

The calendar algebra presented in the previous section can represent all the granularities that are periodical with finite exceptions (i.e., any granularity G such that bottom groups periodically with finite exceptions into G). Since with the periodical representations defined in Section 2 it is not possible to express the finite exceptions, we need to restrict the calendar algebra so that it cannot represent them. This implies allowing the *Subset* operation to be only used as the last step of deriving a granularity. Note that in the calendar algebra presented by Ning et al. (2002) there was an extension to the altering-tick operation to allow the usage of ∞ as the m parameter (i.e., $G' = \mathit{Alter}^{\infty}_{l,k}(G_2, G_1)$); the resulting granularity has a single exception hence is not periodic. This extension is disallowed here in order to generate periodical granularities only (without finite exceptions).

The conversion process can be divided into three steps: in the first one the period length and period label distance are computed; in the second we derive the set $\mathcal{L}^P$ of labels in one period, and in the last one the composition of the explicit granules is computed. For each operation we identify the correct formulas and algorithms for the three steps.

The **first step** consists in computing the period length and the period label distance of the resulting granularity. Those values are calculated as a function of the parameters (e.g. the "grouping factor" m, in the *Group* operation) and the operand granularities (actually their period lengths and period label distances).

The **second step** in the conversion process is the identification of the label set of the resulting granularity. In Section 2.4 we pointed out that in order to fully characterize a granularity it is sufficient to identify the labels in any period of the granularity. In spite of this theoretical result, to perform the computations required by each operation we need the explicit granules of the operand granularities to be "aligned". There are two possible approaches: the first one consist in computing the explicit granules in any period and then recalculate the needed granules in the correct position in order to eventually align them. The second one consists in aligning all the periods containing the explicit granules with a fixed granule in the bottom granularity. After considering both possibilities, for performance reasons, we decided to adopt the second approach. We decided to use $\perp(1)$ as the "alignment point" for all the granularities. A formal definition of the used formalism follows.

Let G be a granularity and i be the smallest positive integer such that $\lceil i \rceil^G$ is defined. We call $l_G = \lceil i \rceil^G$ and $\overline{\mathcal{L}}_G$ the set of labels of G contained in $l_G \dots l_G + N_G - 1$. Note that this definition of $\overline{\mathcal{L}}_G$ is an instance of the definition of $\mathcal{L}^P$ given in Section 2.4. The definition of $\overline{\mathcal{L}}_G$ provided here is useful for representing G and actually the final goal of this step is to compute $\overline{\mathcal{L}}_G$; however $\overline{\mathcal{L}}_G$ is not suitable for performing the computations. The problem is that if $G(l_G)$ starts before $\perp(1)$ (i.e., $min(\lfloor l_G \rfloor^G) < 1$) then the granule $G(l_G + N_G)$ begins at P_G or before P_G, and hence $G(l_G + N_G)$ is necessary for the computations; however $l_G + N_G \notin \overline{\mathcal{L}}_G$.

To solve the problem we introduce the symbol $\hat{\mathcal{L}}_G$ to represent the set of all labels of granules of G that cover one in $\perp(1) \dots \perp(P_G)$. It is easily seen that if $G(l_G)$ does not cover $\perp(0)$, then $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G$, otherwise $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G \cup \{l_G + N_G\}$. Therefore the conversion between $\overline{\mathcal{L}}$ and $\hat{\mathcal{L}}$ and vice versa is immediate.

The notion of $\hat{\mathcal{L}}$ is still not enough to perform the computations. The problem is that when a granularity G is used as an operand in an operation, the period length of the resulting granularity G' is generally bigger than the period length of G. Therefore it is necessary to extend the notion of $\hat{\mathcal{L}}_G$ to the period length $P_{G'}$ of G' using $P_{G'}$ in spite of P_G in the definition of $\hat{\mathcal{L}}$. The symbol used for this notion is $\hat{\mathcal{L}}_G^{P_{G'}}$.

The idea is that when G is used as the operand in an operation that generates G', $\hat{\mathcal{L}}_G^{P_{G'}}$ is computed from $\overline{\mathcal{L}}_G$. This set is then used by the formula that we provide below to compute $\overline{\mathcal{L}}_{G'}$.

The computation of $\overline{\mathcal{L}}_{G'}$ is performed as follows: if G' is defined by an operation that returns a full-integer labeled granularity, then it is sufficient to compute the value of l'_G. Indeed it is easily seen that $\overline{\mathcal{L}}_{G'} = \{i \in \mathbb{Z} | l'_G \leq i \leq l'_G + N_{G'} - 1\}$. If G' is defined by any other algebraic operation, we provide the formulas to compute $\hat{\mathcal{L}}_{G'}$; from $\hat{\mathcal{L}}_{G'}$ we easily derive $\overline{\mathcal{L}}_{G'}$.

Example 3 *Figure 3 shows granularities $\perp$, G and H; it is clear that $P_G = P_H = 4$ and $N_G = N_H = 3$. Moreover, $l_G = l_H = 6$ and therefore $\overline{\mathcal{L}}_G = \overline{\mathcal{L}}_H = \{6, 7\}$. Since $0 \notin \lfloor 6 \rfloor^G$ then $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G$. On the other hand, since $0 \in \lfloor 6 \rfloor^H$, then $\hat{\mathcal{L}}_H = \overline{\mathcal{L}}_H \cup \{6 + 3\}$.*

Suppose that a granularity G' has period length $P_{G'} = 8$; then $\hat{\mathcal{L}}_G^{P'_G} = \{6, 7, 9, 10\}$ and $\hat{\mathcal{L}}_H^{P_{G'}} = \{6, 7, 9, 10, 12\}$.

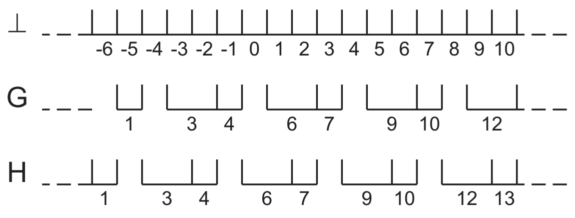


Figure 3: $\overline{\mathcal{L}}$, l, $\hat{\mathcal{L}}$ and $\hat{\mathcal{L}}^{P_{G'}}$ examples

The **third** (and last) **step** of the conversion process is the computation of the composition of the explicit granules. Once $\overline{\mathcal{L}}_{G'}$ has been computed, it is sufficient to apply, for each label of $\overline{\mathcal{L}}_{G'}$ the formulas presented in Chapter 3.

In Sections 4.3 to 4.10 we show, for each calendar algebra operation, how to compute the first and second conversion steps.

4.2 Computability Issues

In some of the formulas presented below it is necessary to compute the set S of labels of a granularity G such that $\forall i \in S\ G(i) \subseteq H(j)$ where H is a granularity and j is a specific label of H. Since $\mathcal{L}_G$ contains an infinite number of labels, it is not possible to check, $\forall i \in \mathcal{L}_G$ if $G(i) \subseteq H(j)$. However it is easily seen that $\forall i \in S\ \exists k$ s.t. $G(\lceil k \rceil^G) \subseteq H(j)$. Therefore $\forall i \in S\ \exists k$ s.t. $G(\lceil k \rceil^G)$ is defined and $k \in \lfloor j \rfloor^H$.

Therefore we compute the set S by considering all the labels i of $\mathcal{L}_G$ s.t. $\exists n \in \lfloor j \rfloor^H$ s.t. $\lceil n \rceil^G = i$ and $G(i) \subseteq H(j)$. Since the set $\lfloor j \rfloor^H$ is finite[3], the computation can be performed in a finite time. The consideration is analogous if S is the set such that $\forall i \in S\ G(i) \supseteq H(j)$ or $\forall i \in S\ (G(i) \cap H(j) \neq \emptyset)$.

4.3 The Group Operation

Proposition 1 *If $G' = Group_m(G)$, then:*

1. $P_{G'} = \frac{P_G \cdot m}{GCD(m,N_G)}$ *and* $N_{G'} = \frac{N_G}{GCD(m,N_G)}$;
2. $l_{G'} = \left(\left\lfloor \frac{l_G - 1}{m} \right\rfloor + 1 \right)$;
3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i \cdot m} G(j)$.

Example 4 *Figure 4 shows an example of the group operation: $G' = Group_3(G)$. Since $P_G = 1$ and $N_G = 1$, then $P_{G'} = 3$ and $N_G = 1$. Moreover, since $\overline{\mathcal{L}}_G = \{-7\}$, then $l_G = -7$ and therefore $l_{G'} = -2$ and $\overline{\mathcal{L}}_{G'} = \{-2\}$. Finally $G'(-2) = G(-8) \cup G(-7) \cup G(-6)$ i.e., $G'(-2) = \bot(0) \cup \bot(1) \cup \bot(2)$.*

3. With the calendar algebra it is not possible to define granularities having granules that maps to an infinite set of time instants.

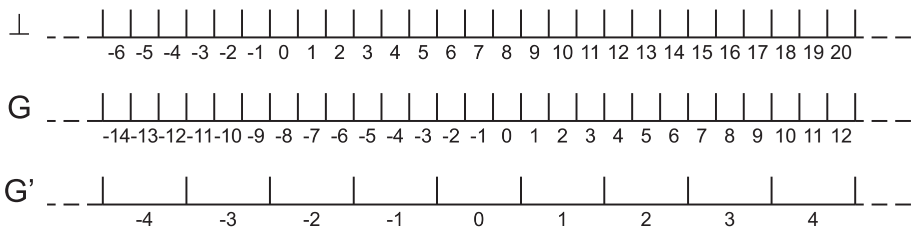


Figure 4: *Group* operation example

4.4 The Altering-tick Operation

Proposition 2 *If $G' = Alter^m_{l,k}(G_2, G_1)$ then:*

1.

$$N_{G'} = lcm\left(N_{G_1}, m, \frac{P_{G_2} \cdot N_{G_1}}{GCD(P_{G_2} \cdot N_{G_1}, P_{G_1})}, \frac{N_{G_2} \cdot m}{GCD(N_{G_2} \cdot m, |k|)}\right)$$

and

$$P_{G'} = \left(\frac{N_{G'} \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{N_{G'} \cdot k}{m}\right) \cdot \frac{P_{G_2}}{N_{G_2}}$$

2. $l_{G'} = \lceil l_{G_2} \rceil^{G'}_{G_2}$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j=b'_i}^{t'_i} G(j)$ *where b'_i and t'_i are defined in Section 3.1.2.*

Referring to step 2., note that when computing $l_{G'}$ the explicit characterization of the granules of G' is still unknown. To perform the operation $\lceil l_{G_2} \rceil^{G'}_{G_2}$ we need to know at least the explicit granules of one of its periods. We choose to compute the granules labeled by $1 \ldots N_{G'}$. When $l_{G'}$ is derived, the granules labeled by $l_{G'} \ldots l_{G'} + N_{G'} - 1$ will be computed so that the explicit granules are aligned to $\perp(1)$ as required.

Example 5 *Figure 5 shows an example of the altering-tick operation: $G' = Alter^3_{2,1}(G_2, G_1)$. Since $P_{G_1} = 4$, $N_{G_1} = 1$, $P_{G_2} = 4$ and $N_{G_2} = 2$, then $N_{G'} = 6$ and $P_{G'} = 28$. Moreover, since $\overline{\mathcal{L}}_{G_2} = \{-10, -9\}$, then $l_{G_2} = -10$ and therefore $l_{G'} = \lceil -10 \rceil^{G'}_{G_2} = -4$ and hence $\overline{\mathcal{L}}_{G_2} = \{-4, -3, \ldots, 0, 1\}$. Finally $G'(-4) = G_1(-11) \cup G_1(-10) \cup G_1(-9) = \perp(-1) \cup \perp(0) \cup \perp(1) \cup \perp(3) \cup \perp(4)$; analogously we derive $G'(-3)$, $G'(-2)$, $G'(-1)$, $G'(0)$ and $G'(1)$.*

4.5 The Shift Operation

Proposition 3 *If $G' = Shift_m(G)$, then:*

1. $P_{G'} = P_{G_1}$ *and* $N_{G'} = N_{G_1}$;

2. $l_{G'} = l_G + m$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = G(i - m)$.

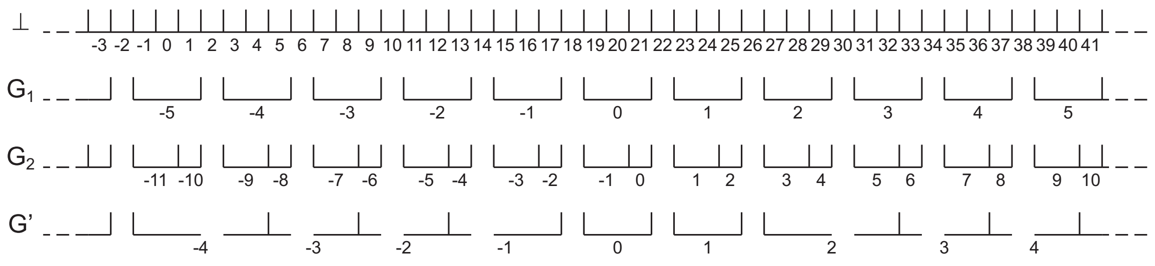


Figure 5: *Alter* operation example

Example 6 *The shifting operation can easily model time differences. Suppose granularity* `USEast-Hour` *stands for the hours of US Eastern Time. Since the hours of the US Pacific Time are 3 hours later than those of US Eastern Time, the hours of US Pacific Time can be generated by* `USPacific-Hour`*=* *Shift*$_{-3}$*(*`USEast-Hour`*).*

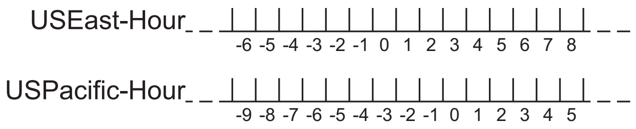


Figure 6: *Shift* operation example

4.6 The Combining Operation

Proposition 4 *Given* $G' = Combining(G_1, G_2)$*, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2})N_{G_1}}{P_{G_1}}$;
2. $\forall i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ *let be* $\widetilde{s}(i) = \{j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$*; then* $\hat{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \widetilde{s}(i) \neq \emptyset\}$*;*
3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j \in s(i)} G_2(j)$.

Example 7 *Figure 7 shows an example of the combining operation:* $G' = \text{Combine}(G_1, G_2)$. *Since* $P_{G_1} = 6$, $N_{G_1} = 2$, $P_{G_2} = 4$ *and* $N_{G_2} = 2$, *then* $P_{G'} = 12$ *and* $N_{G'} = 4$. *Moreover, since* $\overline{\mathcal{L}}_{G_1} = \{1\}$ *and* $0 \in \lfloor 1 \rfloor^{G_1}$, *then* $\hat{\mathcal{L}}_{G_1} = \{1, 3\}$ *and hence* $\hat{\mathcal{L}}_{G_1}^{P_{G'}} = \{1, 3, 5\}$. *Since* $\tilde{s}(i) \neq \emptyset$ *for* $i \in \{1, 3, 5\}$, *then* $\hat{\mathcal{L}}_{G'} = \{1, 3, 5\}$*; moreover, since* $0 \in \lfloor 1 \rfloor^{G'}$, *then* $\overline{\mathcal{L}}_{G'} = \{1, 3\}$. *Finally* $s(1) = \{-1, 0\}$ *and* $s(3) = \{2, 3\}$*; consequently,* $G'(1) = G_2(-1) \cup G_2(0)$ *i.e.,* $G'(1) = \perp(-1) \cup \perp(0) \cup \perp(1)$ *and* $G'(3) = G_2(2) \cup G_2(3)$ *i.e.,* $G'(3) = \perp(4) \cup \perp(5) \cup \perp(7)$.

4.7 The Anchored Grouping Operation

Proposition 5 *Given* $G' = Anchored\text{-}group(G_1, G_2)$*, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_2}}{P_{G_2}}$;

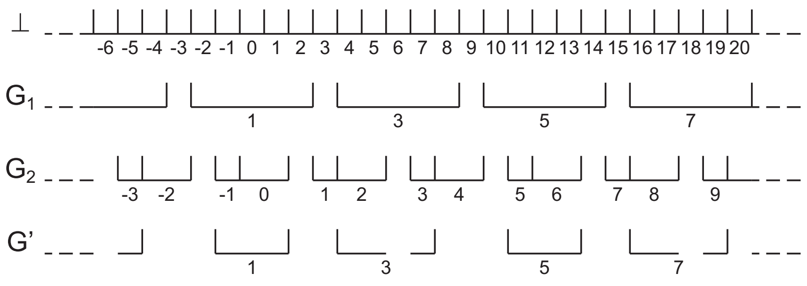


Figure 7: *Combine* operation example

2.

$$\hat{\mathcal{L}}_{G'} = \begin{cases} \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \textit{if } l_{G_2} = l_{G_1}, \\ \{l'_{G_2}\} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \textit{otherwise,} \end{cases}$$

where l'_{G_2} is the greatest among the labels of $\mathcal{L}_{G_2}$ that are smaller than l_{G_2}.

3. $\forall i \in \overline{\mathcal{L}}_{G'}$ $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ where i' is the next label of G_2 after i.

Example 8 *Figure 8 shows an example of the anchored grouping operation: the `USweek` (i.e., a week starting with a `Sunday`) is defined by the operation Anchored-group(`day`, `Sunday`). Since $P_{\texttt{day}} = 1$ and $P_{\texttt{Sunday}} = 7$, then the period length of `USweek` is 7. Moreover since $l_{\texttt{day}} = 11$, $l_{\texttt{Sunday}} = 14$ and $\hat{\mathcal{L}}_{\texttt{Sunday}}^{P_{\texttt{USweek}}} = \{14\}$, then $\hat{\mathcal{L}}_{\texttt{USweek}} = \{7\} \cup \{14\}$. Clearly, since $0 \in \lfloor 7 \rfloor^{\texttt{USweek}}$ then $\overline{\mathcal{L}}_{\texttt{USweek}} = \{7\}$. Finally, $\texttt{USweek}(7) = \bigcup_{j=7}^{13} \texttt{day}(j) = \bigcup_{k=-3}^{3} \bot(k)$.*

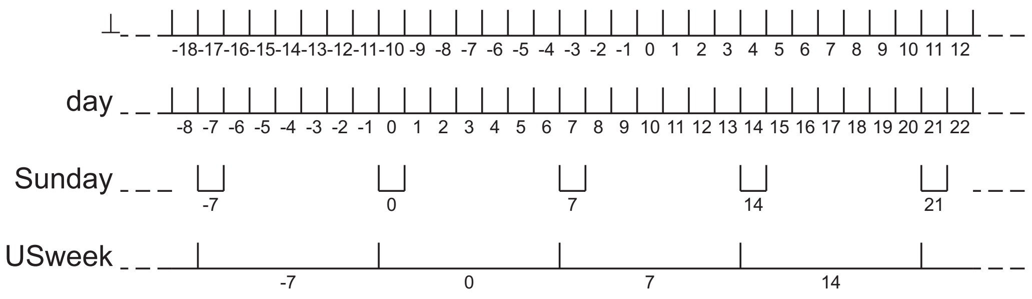


Figure 8: *Anchored Grouping* operation example

4.8 The Subset Operation

The *Subset* operation only modifies the operand granularity by introducing the bounds. The period length, the period label distance, $\overline{\mathcal{L}}$ and the composition of the explicit granules are not affected.

4.9 The Selecting Operations

4.9.1 The Select-down Operation

Proposition 6 *Given $G' = \text{Select-down}^l_k(G_1, G_2)$, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_1}}{P_{G_1}}$;
2. $\forall i \in \mathcal{L}_{G_2}$ *let*
$$A(i) = \Delta^l_k \left(\{j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i)\}\right).$$
Then
$$\hat{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}} \left\{a \in A(i) | a \in \hat{\mathcal{L}}^{P_{G'}}_{G_1}\right\};$$
3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = G_1(i)$.

Example 9 *Figure 9 shows an example of the Select-down operation in which granularity G' is defined as: $G' = \text{Select-down}^1_2(G_1, G_2)$. Since $P_{G_1} = 4$, $N_{G_1} = 2$ and $P_{G_2} = 6$ then $P_{G'} = 12$ and $N_{G'} = 6$. Moreover, since $\overline{\mathcal{L}}_{G_2} = \{-3\}$ and $0 \in \lfloor -3 \rfloor^{G_2}$, then $\hat{\mathcal{L}}_{G_2} = \{-3, -2\}$ and $\hat{\mathcal{L}}^{P_{G'}}_{G_2} = \{-3, -2, -1\}$. Intuitively, $A(-3) = \{-5\}$, $A(-2) = \{-2\}$ and $A(-1) = \{1\}$. Hence $\hat{\mathcal{L}}_{G'} = \{-5, -2, 1\}$ and therefore, since $0 \in \lfloor -5 \rfloor^{G'}$, $\overline{\mathcal{L}}_{G'} = \{-5, -2\}$. Finally $G'(-5) = G_1(-5) = \bot(0) \cup \bot(1)$ and $G'(-2) = G_1(-2) = \bot(6)$.*

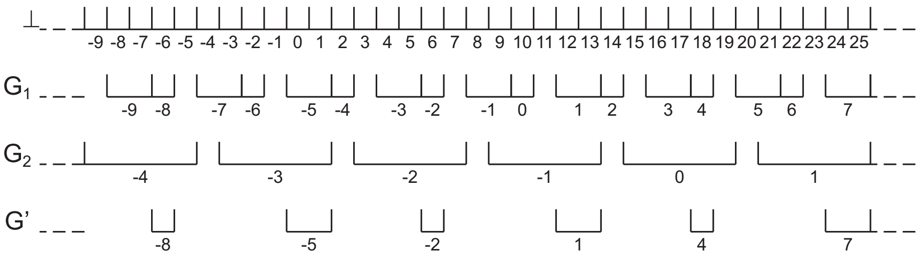


Figure 9: *Select-down* operation example

4.9.2 The Select-up Operation

Proposition 7 *Given $G' = \text{Select-up}(G_1, G_2)$, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_1}}{P_{G_1}}$;
2.
$$\hat{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}^{P_{G'}}_{G_1} | \exists j \in \mathcal{L}_{G_2}\ s.t. \emptyset \neq G_2(j) \subseteq G_1(i)\};$$
3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = G_1(i)$.

Example 10 *Figure 10 shows an example of the Select-up operation: $G' = \text{Select-up}(G_1, G_2)$. Since $P_{G_1} = 6$, $N_{G_1} = 3$ and $P_{G_2} = 4$ then $P_{G'} = 12$ and $N_{G'} = 6$. Moreover, since $\overline{\mathcal{L}}_{G_1} = \{-3, -2, -1\}$ and $0 \in \lfloor -3 \rfloor^{G_2}$, then $\hat{\mathcal{L}}_{G_1} = \{-3, -2, -1, 0\}$ and $\hat{\mathcal{L}}_{G_1}^{P'_G} = \{-3, -2, -1, 0, 1, 2, 3\}$. Since $G_1(-3) \supseteq G_2(-6)$, $G_1(-1) \supseteq G_2(-4)$ and $G_1(3) \supseteq G_2(0)$ then $\hat{\mathcal{L}}_{G'} = \{-3, -1, 3\}$ and, since $0 \in \lfloor -3 \rfloor^{G'}$, then $\overline{\mathcal{L}}_{G'} = \{-3, 1\}$ Finally $G'(-3) = G_1(-3) = \perp(0) \cup \perp(1)$ and $G'(-1) = G_1(-1) = \perp(4)$.*

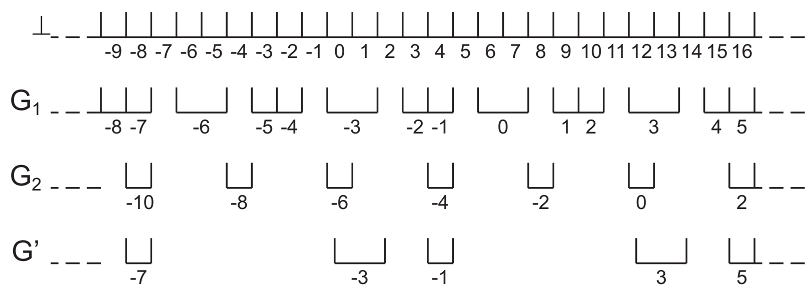


Figure 10: *Select-up* operation example

4.9.3 The Select-by-intersect Operation

Proposition 8 *Given $G' = \text{Select-by-intersect}_k^l(G_1, G_2)$, then:*

1. *$P_{G'} = lcm(P_{G_1}, P_{G_2})$ and $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$;*
2. *then $\forall i \in \mathcal{L}_{G_2}$ let*

$$A(i) = \Delta_k^l \left(\{ j \in \mathcal{L}_{G_1} | G_1(j) \cap G_2(i) \neq \emptyset \} \right).$$

then

$$\hat{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}} \left\{ a \in A(i) | a \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} \right\}.$$

3. *$\forall i \in \overline{\mathcal{L}}_{G'}$ $G'(i) = G_1(i)$.*

Example 11 *Figure 11 shows an example of the Select-by-intersect operation in which $G' = \text{Select-by-intersect}_2^1(G_1, G_2)$. Since $P_{G_1} = 4$, $N_{G_1} = 2$ and $P_{G_2} = 6$ then $P_{G'} = 12$ and $N_{G'} = 6$. Moreover, since $\overline{\mathcal{L}}_{G_2} = \{-3\}$ and $0 \in \lfloor -3 \rfloor^{G_2}$, then $\hat{\mathcal{L}}_{G_2} = \{-3, -2\}$ and $\hat{\mathcal{L}}_{G_2}^{P_{G'}} = \{-3, -2, -1\}$. Intuitively, $A(-3) = \{-6\}$, $A(-2) = \{-2\}$ and $A(-1) = \{0\}$. Hence $\hat{\mathcal{L}}_{G'} = \{-2, 0\}$ and therefore, since $0 \notin \lfloor -5 \rfloor^{G'}$, then $\overline{\mathcal{L}}_{G'} = \{-2, 0\}$. Finally $G'(-2) = G_1(-2) = \perp(6)$ and $G'(0) = G_1(0) = \perp(10)$.*

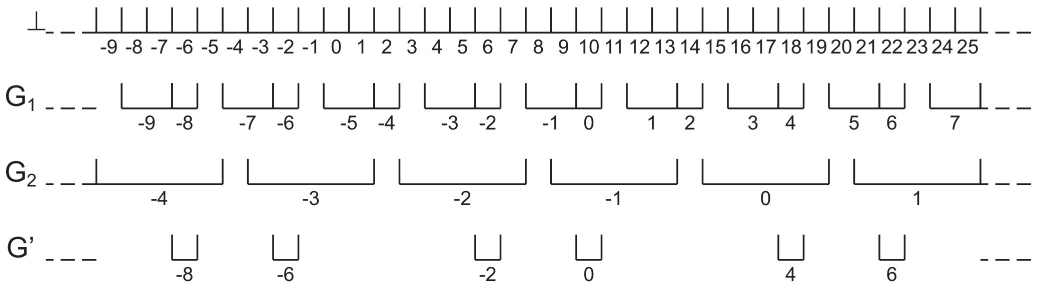


Figure 11: *Select-by-intersect* operation example

4.10 The Set Operations

Since a set operation is valid if the granularities used as argument are both labeled aligned granularity of another granularity, the following property is used.

Proposition 9 *If G is a labeled aligned subgranularity of H, then $\frac{N_G}{P_G} = \frac{N_H}{P_H}$.*

Proposition 10 *Given $G' = G_1 \cup G_2$, $G'' = G_1 \cap G_2$ and $G''' = G_1 \setminus G_2$, then:*

1. $P_{G'} = P_{G''} = P_{G'''} = lcm(P_{G_1}, P_{G_2})$ *and*
   $N_{G'} = N_{G''} = N_{G'''} = \frac{lcm(P_{G_1},P_{G_2})N_{G_1}}{P_{G_1}} = \frac{lcm(P_{G_1},P_{G_2})N_{G_2}}{P_{G_2}}$;

2. $\hat{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G_1}^{P_{G'}} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}$; $\hat{\mathcal{L}}_{G''} = \hat{\mathcal{L}}_{G_1}^{P_{G''}} \cap \hat{\mathcal{L}}_{G_2}^{P_{G''}}$; $\hat{\mathcal{L}}_{G'''} = \hat{\mathcal{L}}_{G_1}^{P_{G'''}} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'''}}$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \begin{cases} G_1(i), & i \in \mathcal{L}_{G_1} \\ G_2(i), & \textit{otherwise}, \end{cases}$

   $\forall i \in \overline{\mathcal{L}}_{G''}\ G''(i) = G_1(i)$ *and* $\forall i \in \overline{\mathcal{L}}_{G'''}\ G'''(i) = G_1(i)$

Example 12 *Figure 12 shows an example of the set operations. Note that both G_1 and G_2 are labeled aligned subgranularities of H. Then $G' = G_1 \cup G_2$, $G'' = G_1 \cap G_2$ and $G''' = G_1 \setminus G_2$. Since $P_{G_1} = P_{G_2} = 6$ and $N_{G_1} = N_{G_2} = 6$ then $P_{G'} = P_{G''} = P_{G'''} = 6$ and $N_{G'} = N_{G''} = N_{G'''} = 2$. Moreover, since $\hat{\mathcal{L}}_{G_1} = \{1, 2\}$ and $\hat{\mathcal{L}}_{G_2} = \{2, 3\}$, then $\hat{\mathcal{L}}_{G'} = \{1, 2, 3\}$, $\hat{\mathcal{L}}_{G''} = \{2\}$ and $\hat{\mathcal{L}}_{G'''} = \{1\}$. Finally $G'(1) = G_1(1)$, $G'(2) = G_1(2)$ and $G'(3) = G_2(3)$; $G''(2) = G_1(2)$ and $G'''(1) = G_1(1)$.*

4.11 Relabeling

Granularity processing algorithms are much simpler if restricted to operate on full-integer labeled granularities. Moreover, a further simplification is obtained by using only the positive integers as the set of labels (i.e., $\mathcal{L} = \mathbb{Z}^+$).

In this section we show how to relabel a granularity G to obtain a full-integer labeled granularity G'. A granularity G'' such that $\mathcal{L}_{G''} = \mathbb{Z}^+$ can be obtained by using $G'' = Subset_1^{\infty}(G')$

Note that with the relabeling process some information is lost: for example, if G is a labeled aligned subgranularity of H and $G \neq H$, then, after the relabeling, G is not a

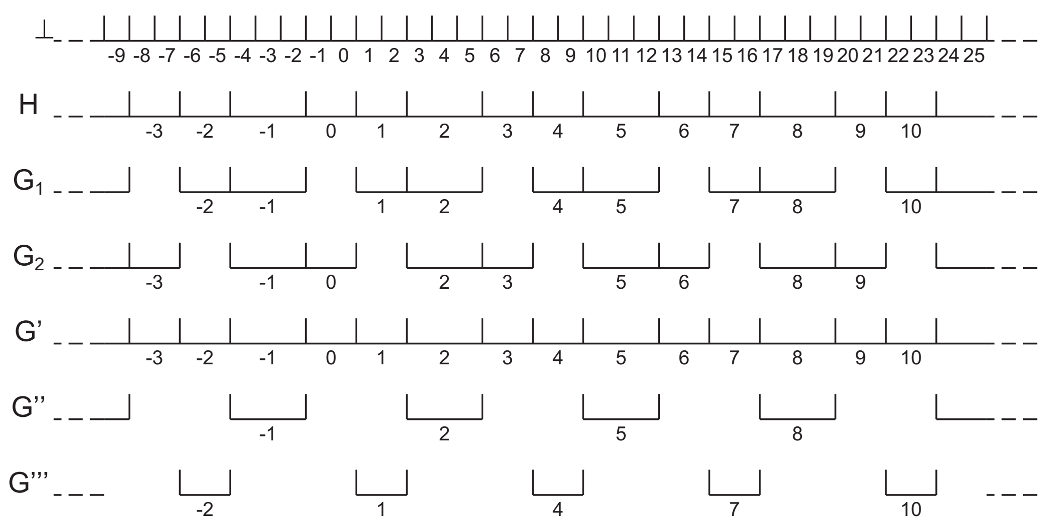


Figure 12: *Set* operations example

labeled aligned subgranularity of H. The lost information is semantically meaningful in the calendar algebra, and therefore the relabeling must be performed only when the granularity will not be used as an operator in an algebraic operation.

Let G be a labeled granularity, i and j integers with $i \in \mathcal{L}_G$ s.t. $G(i) \neq \emptyset$. The relabeling operation $Relabel_i^j(G)$ generates a full-integer labeled granularity G' by relabeling $G(i)$ as $G'(j)$ and relabel the next (and previous) granule of G by the next (and previous, respectively) integer. More formally, for each integer k, if $k = j$, then let $G'(k) = G(i)$, and otherwise let $G'(k) = G(i')$ where $G(i')$ is the $|j-k|$-th granule of G after (before, respectively) $G(i)$. If the required $|j-k|$-th granule of G does not exist, then let $G'(k) = \emptyset$. Note the G' is always a full-integer labeled granularity.

The relabeling procedure can be implemented in the periodic representation we adopted by computing the value of $l_{G'}$. It is easily seen that once $l_{G'}$ is known, the full characterization of G' can be obtained with: $P_{G'} = P_G$; $N_{G'} = R_{G'} = R_G$ and $\overline{\mathcal{L}}_{G'} = \{l_{G'}, l_{G'}+1, \dots, l_{G'}+N_{G'}-2, l_{G'}+N_{G'}-1\}$. It is clear that the explicit representation of the granules is not modified.

To compute $l_{G'}$ consider the label $i' = i - \left\lfloor \frac{i-l_G}{N_G} \right\rfloor \cdot N_G$; i' represents the label of $\overline{\mathcal{L}}_G$ such that $i - i'$ is a multiple of N_G. Therefore it is clear that the label $j' \in \overline{\mathcal{L}}_{G'}$ s.t. $G'(j') = G(i')$ can be computed by $j' = j - \left\lfloor \frac{i-l_G}{N_G} \right\rfloor \cdot N_{G'}$. Finally $l_{G'}$ is obtained with $l_{G'} = j' - |\delta|$ where δ is the distance, in terms of number of granules of G, from $G(l_G)$ to $G(i')$.

Example 13 *Figure 13 shows an example of the Relabel operation: $G' = Relabel_{33}^{4}(G)$. Since $P_G = 4$ and $R_G = 2$ then $P_{G'} = 4$ and $N_{G'} = 2$. Moreover, $i' = 33 - \lfloor\frac{33-6}{5}\rfloor \cdot 5 = 8$ and $j' = 4 - \lfloor\frac{33-6}{5}\rfloor \cdot 2 = -6$. Since $l_G = 6$ and $i' = 8$ then $G(i')$ is the next granule of G after $G(l_G)$. Then $\delta = 1$ and hence $l_{G'} = -6 - 1 = -7$. It follows that $\overline{\mathcal{L}}_{G'} = \{-7, -6\}$. Finally $G'(-7) = G(6)$ and $G'(-6) = G(8)$.*

The GSTP constraint solver imposes that the first non-empty granule of any granularity ($\perp$ included) is labeled with 1. Therefore, when using the relabeling operation for producing

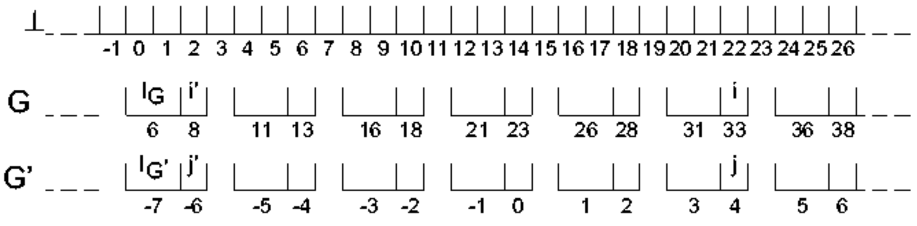


Figure 13: *Relabeling* example

granularities for GSTP, the parameter j must be set to 1. The parameter i has to be equal to the smallest label among those that identify granules of G covering granules of $\perp$ that are all labeled with positive values. By definition of l_G, $i = l_G$ if $min(\lfloor l_G \rfloor^G) > 0$; otherwise i is the next label of G after l_G.

4.12 Complexity Issues

For each operation the time necessary to perform the three conversion steps, depends on the operation parameters (e.g. the "grouping factor" m, in the *Group* operation) and on the operand granularities (in particular the period length, the period label distance and the number of granules in one period).

A central issue is that if an operand granularity is not the bottom granularity, then its period is a function of the periods of the granularities that are the operands in the operation that defines it. For most of the algebraic operations, in the worst case the period of the resulting granularity is the product of the periods of the operands granularity.

For all operations, the **first step** in the conversion process can be performed in a constant or logarithmic time. Indeed the formulas necessary to derive the period length and the period label distance involve (i) standard arithmetic operations, (ii) the computation of the Greatest Common Divisor and (iii) the computation of the least common multiple. Part (i) can be computed in a constant time while (ii) and (iii) can be computed in a logarithmic time using Euclid's algorithm.

For some operations, the **second step** can be performed in constant time (e.g. *Group*, *Shift* or *Anchored-group*) or in linear time (e.g. set operations). For the other operations it is necessary to compute the set S of labels of a granularity G such that $\forall i \in S\ G(i) \subseteq H(j)$ where H is a granularity and $j \in \mathcal{L}_H$ (analogously if S is the set such that $\forall i \in S\ G(i) \supseteq H(j)$ or $\forall i \in S\ (G(i) \cap H(j) \neq \emptyset)$). This computation needs to be performed once for each granule $i \in P_H^{P_{G'}}$. The idea of the algorithm for solving the problem has been presented in Section 4.2. Several optimizations can be applied to that algorithm, but in the worst case (when H covers the entire time domain) it is necessary to perform a number of $\lceil \cdot \rceil^G$ operations linear in the period length of the resulting granularity. If an optimized data structure is used to represent the granularities, the $\lceil \cdot \rceil^G$ operation can be performed in constant time [4], then the time necessary to perform the second step is linear in the period length of the resulting granularity ($O(P_{G'})$).

The **last step** in the conversion process is performed in linear time with respect to the number of granules in a period of G'.

4. If a non-optimized data structure is used, $\lceil \cdot \rceil^G$ requires logarithmic time.

The complexity analysis of the conversion of a general algebraic expression needs to consider the composition of the operations and hence their complexity. Finally, relabeling, can be done in linear time.

A more detailed complexity analysis is out of the scope of this work.

5. Minimal Representation and Experimental Results

In this section we address the problem of guaranteeing that the converted representation is minimal in terms of the period length. As we will show in Example 14 the conversion formulas proposed in this paper do not guarantee a minimal representation of the result and it is not clear if conversion formulas ensuring minimality exist. Our approach is to apply a minimization step in the conversion.

The practical applicability of the minimization step depends on the period length of the representation that is to be minimized. Indeed, in our tests we noted that the minimization step is efficient if the conversion formulas proposed in Section 4 are adopted, while it is impractical when the conversion procedure returns a period that is orders of magnitude higher than the minimal one as would be the case if conversion formulas were constructed in a naive way.

5.1 Period Length Minimization

As stated in Section 2, each granularity can have different periodical representations and, for a given granularity, it is possible to identify a set of representations that are *minimal* i.e. adopting the smallest period length.

Unfortunately, the conversions do not always return a minimal representation, as shown by Example 14.

Example 14 *Consider a calendar that has* `day` *as the bottom granularity. We can define* `week` *as* $\texttt{week} = Group_7(\texttt{day})$*; by applying the formulas for the Group operation we obtain* $P_{\texttt{week}} = 7$ *and* $N_{\texttt{week}} = 1$.

We can now apply the Altering-tick operation to add one day to every first week every two weeks. Let this granularity be $G_1 = Alter^{2}_{1,1}(\texttt{day}, \texttt{week})$*; applying the formulas for the Altering-tick operation we obtain* $P_{G_1} = 15$ *and* $N_{G_1} = 2$.

We can again apply the Altering-tick operation to create a granularity G_2 *by removing one day from every first granule of* G_1 *every two granules of* G_1*:* $G_2 = Alter^{2}_{1,-1}(\texttt{day}, G_1)$*. Intuitively, by applying this operation we should get back to the granularity* `week`*, however using the formulas for the Altering-tick operation we obtain* $P_{G_2} = 14$ *and* $N_{G_2} = 2$*; Hence* G_2 *is not minimal.*

In order to qualitatively evaluate how close to the minimal representations the results of our conversions are, we performed a set of tests using an algorithm (Bettini & Mascetti, 2005) for minimality checking. In our experimental results the conversions of algebraic expressions defining granularities in real-world calendars, including many user-defined non-standard ones, always returned exactly minimal representations. Non-minimal ones could only be obtained by artificial examples like the one presented in Example 14.

Although a non-minimal result is unlikely in practical calendars, the minimality of the granularity representation is known to greatly affect the performance of the algorithms for granularity processing, e.g., granularity constraint processing (Bettini et al., 2002a), calendar calculations (Urgun et al., 2007), workflow temporal support (Combi & Pozzi, 2003). Hence, we considered an extension of the conversion algorithm by adding a minimization step exploiting the technique illustrated by Bettini et al. (2005) to derive a minimal representation.

The choice of using only the conversion algorithm or the extended one with minimizations, should probably be driven by performance considerations. In Section 5.3 we report the results of our experiments showing that generally it is advantageous to apply the minimization step. In our implementation, presented in Section 5.2, it is possible to specify if the minimization step should be performed.

5.2 Implementation of the *CalendarConverter* Web Service

The conversion formulas presented in Section 4 have been implemented into the *CalendarConverter* web service that converts Calendar Algebra representations into the equivalent periodical ones. More precisely, given a calendar in which granularities are expressed by Calendar Algebra operations, the service converts each operation into an equivalent periodical representation.

The service first rewrites each calendar algebra expression in order to express it only in terms of the bottom granularity. For example, if the bottom granularity is `hour`, the expression $\texttt{Monday} = \textit{Select-down}_1^1(\texttt{day}, \texttt{week})$ is changed to

$$\texttt{Monday} = \textit{Select-down}_1^1(\textit{Group}_{24}(\texttt{hour}), \textit{Group}_7(\textit{Group}_{24}(\texttt{hour})))$$

Then, Procedure 1 is run for each granularity's expression. The idea is that the periodical representation of each subexpression is recursively computed starting from the expressions having the bottom granularity as operand. Once each operand of a given operation has been converted to periodical representation, the corresponding formula presented in Section 4 is applied. We call this step the *ConvertOperation* procedure.

A trivial optimization of Procedure 1 consists in caching the results of the conversions of each subexpression so that it is computed only once, even if the subexpression appears several times (like $\textit{Group}_{24}(\texttt{hour})$ in the above `Monday` definition).

5.3 Experimental Results

Our experiments address two main issues: first, we evaluate how the conversion formulas impact on the practical applicability of the conversion procedure and, second, we evaluate how useful is the minimization step.

For the first issue, we execute the conversion procedure with two different sets of conversion formulas and compare the results. The first set is laid out in Section 4. The other, that is less optimized, is taken from the preliminary version of this paper (Bettini et al., 2004).

Table 1 shows that when converting calendars having granularities with small minimal period length (first two rows), using the formulas in Section 4 improves the performance by one order of magnitude; However, conversions and minimizations are almost instantaneous with both approaches. On the contrary, when the minimal period length is higher,

Procedure 1 ConvertExpression

- **Input**: a calendar algebra expression *ex*; a boolean value *minimize* that is set to **true** if the minimization step is to be executed;
- **Output**: the periodical representation of *ex*;
- **Method**:

```
1: if (ex is the bottom granularity) then
2:     return the periodical representation of the bottom granularity
3: end if
4: operands := $\emptyset$
5: for (each operand op of ex) do
6:     add ConvertExpression(op, minimize) to operands;
7: end for
8: result :=ConvertOperation(ex.getOperator(), operands)
9: if (minimize) then
10:    minimize the periodical representation of result
11: end if
12: return result;
```

Table 1: Impact of the conversion formulas on the performance of the conversion and minimization procedures (time in milliseconds).

Calendar		Section 4 formulas			Less optimized formulas		
Period	Bot	Conv.	Min.	Tot.	Conv.	Min.	Tot.
1 year	day	4	2	6	62	32	94
4 years	day	7	2	9	76	55	131
1 year	hour	9	2	11	2,244	126,904	129,148
4 years	hour	16	4	20	4,362	908,504	912,866
100 years	day	127	9	136	3,764	1,434,524	1,438,288

(last three rows) the time required to minimize the periodical representation is up to five orders of magnitude larger if the formulas proposed by Bettini et al. (2004) are used; as a consequence, the entire conversion may require several minutes while, using the formulas presented in Section 4, it still requires only a fraction of a second. If the period length is even larger, the conversion procedure is impractical if the formulas presented by Bettini et al. (2004) are used, and indeed in our experiments we did not obtain a result in less than thirteen hours.

For the second issue, we perform a set of three experiments. In the first one we compare the performance of the conversion procedure with the performance of the minimization step.

In the experiment we consider the case in which the conversion procedure produces minimal representations. In this case the minimization step is always an overhead since it cannot improve the performance of the conversion procedure.

Figure 14 shows the result of the experiment. Four calendars are considered, each one containing a set of granularities of the Gregorian calendar. The four calendars differs in the values of two parameters: the bottom granularity (it is `second` for cal-1 and cal-3 while it is `minute` for cal-2 and cal-4) and the period in which leap years and leap years exceptions are represented (it is 1, 4, 100 and 400 years for cal-1, cal-3, cal-2 and cal-4 respectively); As a consequence, the minimal period length of the granularities `month` and `year` is about $3 \cdot 10^7$ for cal-1, $5 \cdot 10^7$ for cal-2, 10^8 for cal-3 and $2 \cdot 10^8$ for cal-4.

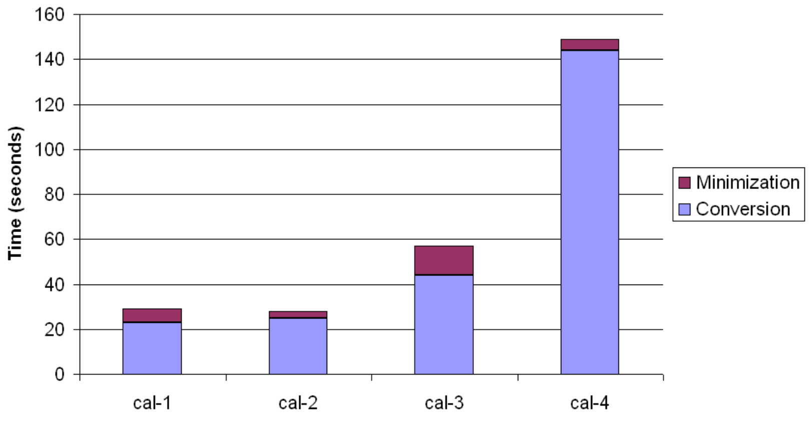


Figure 14: Impact of minimization over conversion; minimal conversions case.

As can be observed in Figure 14, the ratio between the time required to perform the conversions and the time required for the minimization step varies significantly from a minimum of 3% for cal-4 to a maximum of 23% for cal-3. The reason is that the complexity of the conversion procedure is mainly affected by the period length of the granularity having the largest period length. On the other hand, the complexity of the minimization step is affected also by other features of the granularities such as their internal structure and the number of integers that can divide at the same time the period label distance, the period length and the number of granules in one period; For more details see (Bettini & Mascetti, 2005).

In the second experiment we consider the case in which the conversion procedure produces a non-minimal representation for a granularity in the input calendar; in this case it is possible to benefit from the minimization step. For example, suppose that a granularity G is converted and that it is then used as an argument of another Calendar Algebra operation that defines a granularity H. The time required to compute the periodical representation of H strongly depends on the period length of G; If the period length of G is reduced by the execution of the minimization step, the conversion of H can be executed faster.

We produced this situation using a technique similar to the one of Example 14; we created Calendar Algebra definitions of the Gregorian calendar in which the granularity `day` is converted into a granularity having a non-minimal representation. Figure 15 shows the performance obtained converting the same granularities that were used in Figure 14.

The difference was that in this case the definition of the granularity `day` is such that, after the conversion procedure, its period is twice as large as the minimal one (i.e., 48 hours or 2880 minutes or 172800 seconds depending on the bottom granularity that is used). It can be easily seen that in this case the use of the minimization step can improve the performance of the entire algorithm. Indeed, when the minimization step is performed, the conversion procedure requires about one half of the time that is required when no minimization is performed.

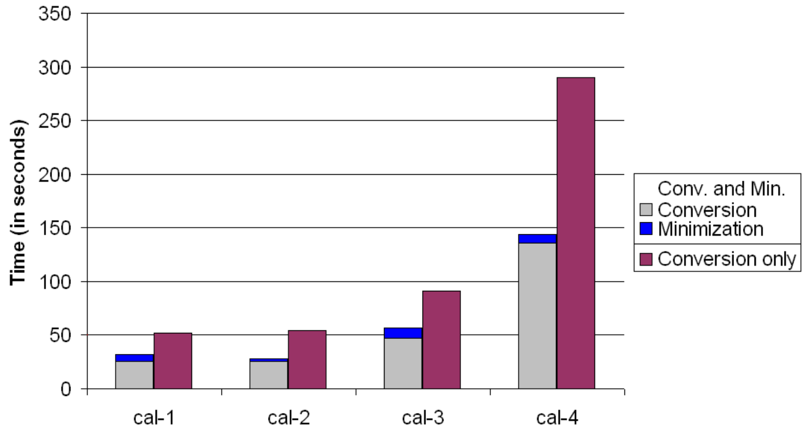


Figure 15: Impact of minimization over conversion; non-minimal case.

In the third experiment we evaluate the impact of the minimal representation on the performance of applications involving intensive manipulations of granularities. In the test we use the GSTP solver as such an application; it computes solutions of temporal constraints with granularities. A description of the architecture of the GSTP system is provided in Section 6.1.

Figure 16 shows our experiments performed on four temporal constraint networks with granularities. The four networks differs in the number of variables, in the number of constraints and in the granularities used to express the constraints. The networks labeled as "non-minimal" use granularities definitions that are obtained with a technique similar to the one used in Example 14, and have a period that is twice as large as the minimal one.

Figure 16 shows that the use of minimal representations greatly improves the performance of the GSTP solver. Indeed in our experiments the ratio between the time required to solve the network using a non-minimal representation and a minimal one is between three and five. Moreover, the more time required to solve the network, the greater the improvement obtained using the minimal representation; this means that for very complex temporal networks we expect the improvement to be even higher.

Considering the results of our experiments, we conclude that, in general, it is advisable to perform the minimization step. In particular, it is very advantageous in the specific case of GSTP, based on the following considerations: i) the time required to perform the minimization step is only a fraction of the time required to perform the conversion procedure, ii) the conversions are performed off-line in most cases, with respect to granularity processing, and conversion results are cached for future use, and iii) the period length strongly influ-

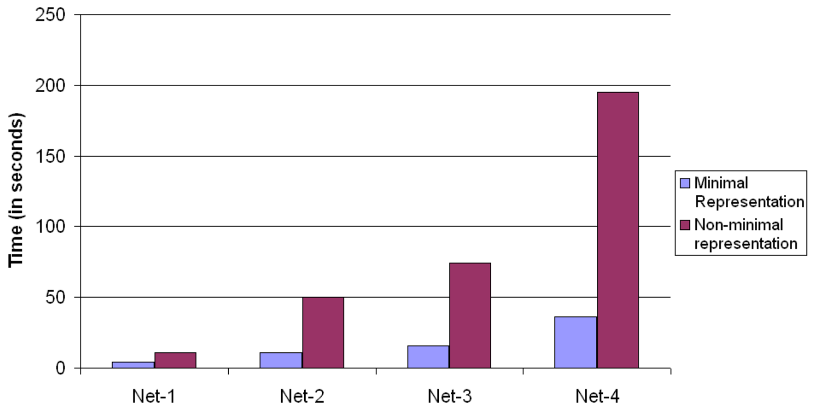


Figure 16: Impact of minimal representations on the performance of the GSTP solver.

ences the GSTP processing time that is in most cases much longer than the time needed for conversion.

6. Applications

In this section we complement the motivations for this work with a sketch of the applications enabled by the proposed conversion. Firstly we describe the GSTP system, as an example of applications involving intensive manipulation of time granularities. GSTP is used to check the consistency and to find solutions of temporal constraint satisfaction problems with granularities[5]; It has also been applied to check the consistency of inter-organizational workflow models (Bettini, Wang, & Jajodia, 2002b). Then, we discuss the use of Calendar Algebra to define new granularities that may later be part of the input of reasoning services, such as GSTP.

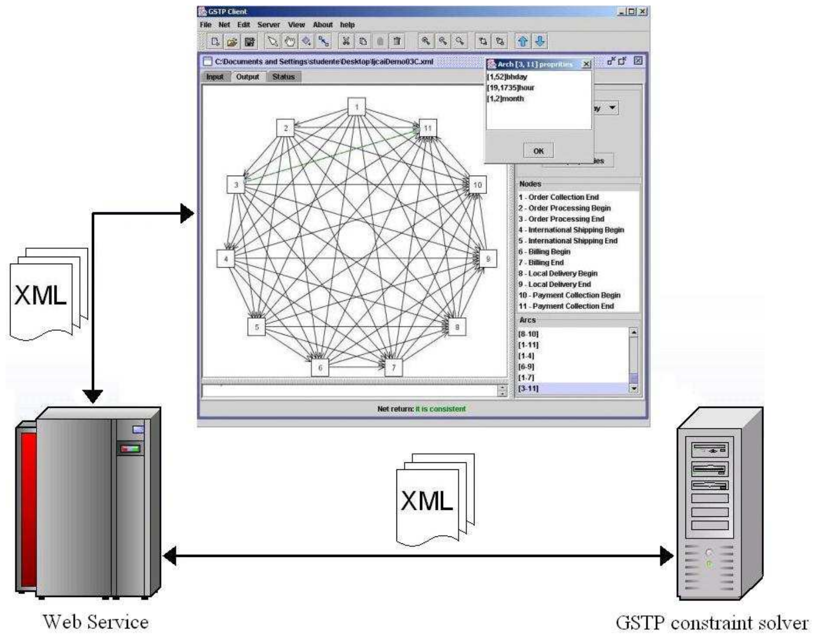
6.1 The GSTP System

The GSTP system has been developed at the University of Milan with the objective of providing universal access to the implementation of a set of algorithms for multi-granularity temporal constraint satisfaction (Bettini et al., 2002a). It allows the user to specify binary constraints of the form $Y - X \in [m, n]G$ where m and n are the minimum and maximum values of the distance from Y to X in terms of granularity G. Variables take values in the positive integers, and unary constraints can be applied on their domains. For example, the constraint: *Event2 should occur 2 to 4 business days after the occurrence of Event1* can be modeled by $Occ_{E2} - Occ_{E1} \in [2, 4]BDay$. This problem is considered an extension of STP (Dechter, Meiri, & Pearl, 1991) to multiple and arbitrary granularities. To our knowledge, GSTP is the only available system to solve this class of temporal constraint satisfaction problems.

Figure 17 shows the general architecture of the GSTP system. There are three main modules: the constraint solver; the web service, which enables external access to the solver;

5. For a detailed description of the system, see (Bettini et al., 2005).

and a user interface that can be used locally or remotely to design and analyze constraint networks.


GSTP Client
File Net Edit Server View About help
C:\Documents and Settings\studente\Desktop\ijcaiDemo03C.xml
Input Output Status
Arch [3, 11] proprities
OK
Nodes
Arcs
Net return: it is consistent
XML
XML
Web Service
GSTP constraint solver

[1,52]bhday
[19,1735]hour
[1,2]month

1 - Order Collection End
2 - Order Processing Begin
3 - Order Processing End
4 - International Shipping Begin
5 - International Shipping End
6 - Billing Begin
7 - Billing End
8 - Local Delivery Begin
9 - Local Delivery End
10 - Payment Collection Begin
11 - Payment Collection End

[8-10]
[1-11]
[1-4]
[6-9]
[1-7]
[3-11]

Figure 17: The GSTP Architecture

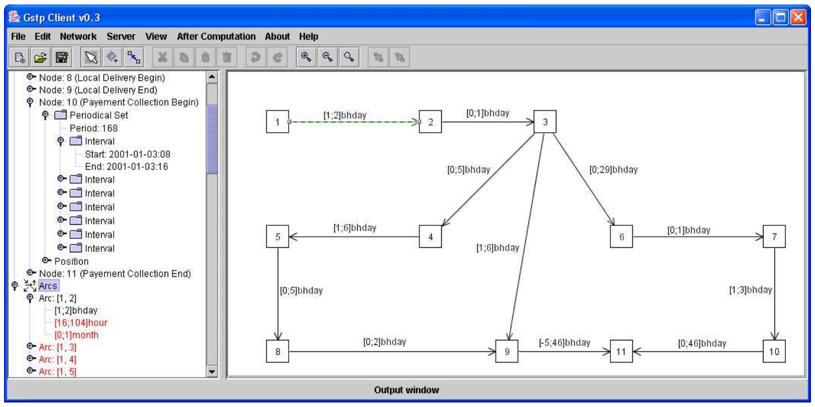
The constraint solver is the C implementation of the ACG algorithm which has been proposed by Bettini et al. (2002a), and it runs on a server machine. Following the approach of Bettini et al. (2002a), the solver uses the representation of granularities based on periodical sets. This representation makes it possible to efficiently compute the core operations on granularities that are required to solve the constraint satisfaction problem. These operations involve, for example, the union and the intersection of periodical sets. While we cannot exclude that these operations may be computed in terms of alternative low level representations, it seems much harder to obtain similar results if a high level representation, such as Calendar Algebra, is used.

The second module of the system is the Web Service that defines, through a WSDL specification, the parameters that can be passed to the constraint solver, including the XML schema for the constraint network specification.

The third module is a remote Java-based user interface, which allows the user to easily edit constraint networks, to submit them to the constraint solver, and to analyze results. In particular, it is possible to have views in terms of specific granularities, to visualize implicit constraints, to browse descriptions of domains, and to obtain a network solution. Fig. 18 shows a screenshot from the interface.

6.2 Defining New Granularities

While the GSTP solver can handle arbitrary granularities, new granularities must be added by editing their explicit periodical representation. This is true in general for any multi-



```
Node: 8 (Local Delivery Begin)
Node: 9 (Local Delivery End)
Node: 10 (Payement Collection Begin)
  Periodical Set
    Period: 168
    Interval
      Start: 2001-01-03:08
      End: 2001-01-03:16
    Interval
    Interval
    Interval
    Interval
    Interval
    Interval
  Position
Node: 11 (Payement Collection End)
Arcs
  Arc: [1, 2]
    [1;2]bhday
    [16;104]hour
    [0;1]month
  Arc: [1, 3]
  Arc: [1, 4]
  Arc: [1, 5]
```

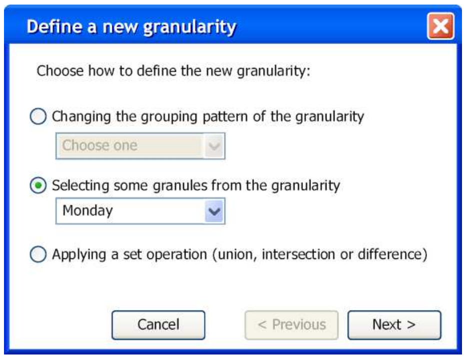
Figure 18: The GSTP User Interface

granularity reasoning service based on a low-level representation of granularities, and it is a painful task when the granularities have a large period. For example, in the experimental results illustrated in Figure 16, we used a representation of the granularity `month` that considers leap years and leap years exceptions in a period of 400 years. In this case, the users have to specify the representation of 4800 granules i.e., the number of months in 400 years.

Because the period length of real world granularities is generally high, a graphical interface does not help if it only supports the user to individually select the explicit granules. An effective solution requires the use of implicit or explicit operations on granules. Among the various proposals, Calendar Algebra provides the richest set of such operators. A question arises: is the definition of granularities in terms of Calendar Algebra really simpler than the specification of the periodical representation? Calendar Algebra does not seem to be user friendly: the exact semantics of each operator may not be immediate for an inexperienced user and some time is required in order to learn how to use each operator.

In practice, we do not think that it is reasonable to ask an unexperienced user to define granularities by writing Calendar Algebra expressions. Nevertheless, we do think that Calendar Algebra can be used by specialized user interfaces to guide the user when specifying granularities. In this sense, we believe that Calendar Algebra plays the same role that SQL does in the definition of databases queries. Similarly to Calendar Algebra, SQL is an abstraction tool that can be directly exploited in all its expressive power by an advanced user, but can also be used by a less experienced user through a graphical user interface, possibly with a reduced expressiveness.

As mentioned above, in the case of periodical representations, graphical user interfaces are not sufficient for making the specification of new granularities practical. On the contrary, in the case of Calendar Algebra, user interfaces can strongly enhance the usability of Calendar Algebra, making its practical use possible also for the definition of involved granularities. There are at least two reasons for this difference. Firstly, the main difficulty of Calendar Algebra is the understanding of the semantics of the operators and the choice of the most appropriate one for a given task. An effective user interface can hide the existence of the algebraic operators to the user showing only how the operators modify existing



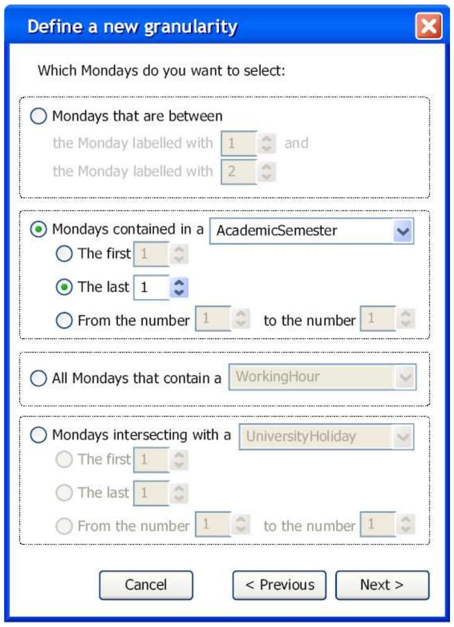
Choose how to define the new granularity:

Changing the grouping pattern of the granularity: Choose one

[✓ Selecting some granules from the granularity]: Monday

Applying a set operation (union, intersection or difference)

(a) Step 1.



Which Mondays do you want to select:

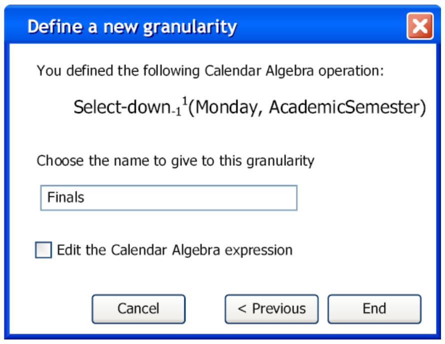
Mondays that are between the Monday labelled with 1 and the Monday labelled with 2

[✓ Mondays contained in a] AcademicSemester
- The first 1
- [✓ The last] 1
- From the number 1 to the number 1

All Mondays that contain a WorkingHour

Mondays intersecting with a UniversityHoliday
- The first 1
- The last 1
- From the number 1 to the number 1

(b) Step 2.



You defined the following Calendar Algebra operation:

Select-down$_{-1}^{1}$(Monday, AcademicSemester)

Choose the name to give to this granularity: Finals

Edit the Calendar Algebra expression

(c) Step 3.

Figure 19: A 3-steps wizard for visually defining a granularity using Calendar Algebra

granularities (i.e., the semantics of the operators). Secondarily, Calendar Algebra allows the compact definition of granularities. This is due to the fact that the Calendar Algebra operations are specifically designed to reflect the intuitive ways in which users define new granularities.

Example 15 shows how a graphical user interface can be effectively used to define a new granularity in terms of Calendar Algebra expression.

Example 15 *This example shows how a graphical user interface can be used to support the user in the definition of the granularity* `final` *as the set of days, each one corresponding to the last Monday of every academic semester. We assume that the granularities* `Monday` *and* `academicSemester` *have already been defined. The graphical user interface that we use in this example is a wizard that guides the user step by step. In the first step (Figure 19(a)) the user chooses the kind of operation he wants to perform. In the second step (Figure 19(b))* *the user can provide more details about how he wants to modify the operand granularity (`Monday`, in the example). The results of this choice is a Calendar Algebra expression that is shown in the third step (Figure 19(c)); in this last window the user can also give a name to the granularity that has been defined.*

6.3 The Global Architecture

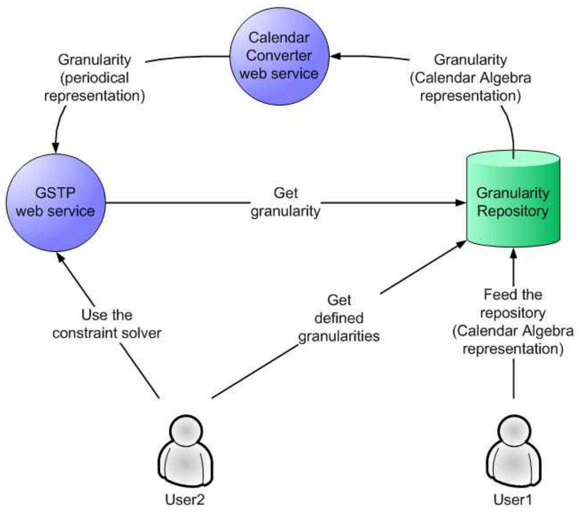

Calendar Converter web service
Granularity (periodical representation)
Granularity (Calendar Algebra representation)
GSTP web service
Get granularity
Granularity Repository
Use the constraint solver
Get defined granularities
Feed the repository (Calendar Algebra representation)
User2
User1

Figure 20: Integration of GSTP and CalendarConverter web services

Figure 20 shows a possible architecture for the integration of GSTP, the interface for new granularity definitions and the CalendarConverter web service. A granularity repository collects the Calendar Algebra definitions. Upon request by the GSTP system definitions are converted in low-level representation by the CalendarConverter web service to be efficiently processed. Clearly, caching techniques can be used to optimize the process.

7. Related Work

Several formalisms have been proposed for symbolic representation of granularities and periodicity. Periodicity and its application in the AI and DB area have been extensively investigated (Tuzhilin & Clifford, 1995; Morris, Shoaff, & Khatib, 1996; Kabanza, Stevenne, & Wolper, 1990; Ladkin, 1986). Regarding symbolic representation, it is well known the formalism proposed by Leban et al. (1986), that is based on the notion of *collection*, and it is intended to represent temporal expressions occurring in natural language. A *collection* is a structured set of time intervals where the order of the collection gives a measure of the structure depth: an order 1 collection is an ordered list of intervals, and an order n ($n > 1$) collection is an ordered list of collections having order $n - 1$. Two operators, called *slicing* and *dicing* are used to operate on collections by selecting specific intervals or sub-collections, and by further dividing an interval into a collection, respectively. For example, `Weeks:during:January2006` divides the interval corresponding to `January2006` into the intervals corresponding to the weeks that are fully contained in that month. This formalism has been adopted with some extensions by many researchers in the AI (Koomen, 1991; Cukierman & Delgrande, 1998) and Database area (Chandra, Segev, & Stonebraker, 1994; Terenziani, 2003). In particular, the control statements `if-then-else` and `while` have been introduced by Chandra et al. (1994) to facilitate the representation of certain sets of intervals. For example, it is possible to specify: *the fourth Saturday of April if not an holiday, and the previous business day otherwise.*

As for the deductive database community, a second influential proposal is the *slice* formalism introduced by Niezette et al. (1992). A slice denotes a (finite or infinite) set of not necessarily consecutive time intervals. For example, the slice `all.Years + {2,4}.Months + {1}.Days` $\triangleright$ `2.Days` denotes a set of intervals corresponding to the first 2 days of February and April of each year.

A totally different approach is the *calendar algebra* described by Ning et al. (2002), and considered in this paper. The representation is based on a rich set of algebraic operators on periodic sets as opposed to *slicing* and *dicing* over nonconvex intervals.

None of the above cited papers provide a mapping to identify how each operator changes the mathematical characterization of the periodicity of the argument expressions. The problem of finding these mappings is not trivial for some operators.

In (Bettini & Sibi, 2000) the expressive power of the algebras proposed by Leban et al. (1986) and Niezette et al. (1992) is compared and an extension to the first is proposed in order to capture a larger set of granularities. Since the periodical representation is used to compare expressiveness, a mapping from calendar expressions in those formalisms to periodical representations can be found in the proofs of that paper. However, since minimality is not an issue for the purpose of comparing expressiveness, in many cases the mapping returns non-minimal representations.

Regarding alternative approaches for low-level representation, we already mentioned that the ones based on strings (Wijsen, 2000) and automata (Dal Lago, Montanari, & Puppis, 2003; Bresolin et al., 2004) may be considered as an alternative for the target of our conversion. As a matter of fact, an example of the conversion of a Calendar Algebra expression into a string based representation can be found in (Dal Lago & Montanari, 2001). A complete conversion procedure appeared during the revision process of this paper in the PhD Dissertation by Puppis (2006). The aim of the conversion is to prove that the *granspecs* formalism, used to represent granularities in terms of automata, has at least the same expressiveness as the Calendar Algebra. Hence, obtaining minimal representations was not the goal. Moreover, in their case minimization is not in terms of the period length, but in terms of the automaton size and automaton complexity. About the complexity of reasoning, given an automaton M, the worst case time complexity of the operations analogous to our *up* and *down* depends linearly on $||M||$, a value computed from M itself and called *complexity of* M. In this sense $||M||$ has the same role of our period length (P), even if a precise relationship between the two values is hard to obtain. In our approach we compute *up* in logarithmic time with respect to P and *down* in linear time with respect to the dimension of the result (that is bounded by P). Other operations, like checking for equivalence, seem to be more complex using automata (Bresolin et al., 2004). Techniques for minimization in terms of automaton complexity are presented by Dal Lago et al. (2003), and the time complexity is proved to be polynomial, even if the exact bound is not explicitly given. In our approach, the worst case time complexity for the minimization is $O(P^{\frac{3}{2}})$ (Bettini &

Mascetti, 2005). Overall, the automata approach is very elegant and well-founded, but, on one side it still misses an implementation in order to have some experimental data to compare with, and on the other side only basic operations have been currently defined; it would be interesting to investigate the definition on that formalism of more complex operations like the ones required by GSTP.

8. Conclusion and Future Work

We have presented an hybrid algorithm that interleaves the conversion of Calendar Algebra subexpressions into periodical sets with the minimization of the period length. We have proved that the algorithm returns set-based granularity representations having minimal period length, which is extremely important for the efficiency of operations on granularities. Based on the technical contribution of this paper, a software system is being developed allowing users to access multi-granularity reasoning services by defining arbitrary time granularities with a high-level formalism. Our current efforts are mainly devoted to completing and refining the development of the different modules of the architecture shown in Section 6.3.

As a future work, we intend to develop effective graphical user interfaces to support the definition of Calendar Algebra expressions in a user friendly way. Example 15 described one of the possible interfaces. Another open issue is how to convert a periodical representation of a granularity into a "user friendly" Calendar Algebra expression. This conversion could be useful, for example, to present the result of a computation performed using the periodical representation. However, a naive conversion may not be effective since the resulting calendar algebra expression could be as involved as the periodical representation from which it is derived. For example, a conversion procedure is presented by Bettini et al. (2000) to prove that the Calendar Algebra is at least as expressive as the periodical representation; however, the resulting Calendar Algebra expression is composed by a number of Calendar Algebra operations that is linear in the number of granules that are in one period of the original granularity. On the contrary, an effective conversion should generate Calendar Algebra expressions that are compact and easily readable by the user. This problem is somehow related to the discovery of calendar-based association rules (Li, Ning, Wang, & Jajodia, 2001). Finally, we intend to investigate the usage of the automaton-based representation as a low-level granularity formalism. It would be interesting to know whether, using this representation, it is possible to compute the same operations that can be computed with the periodical representation and if any performance gain could be achieved.

Acknowledgments

We thank the anonymous referees for their useful comments and suggestions. The work of Bettini and Mascetti was partially supported by Italian MIUR InterLink project N.II04C0EC1D. The work of Wang was partially supported by the US NSF grant IIS-0415023.

Appendix A. Proofs

A.1 Transitivity of the *Periodically Groups Into* Relationship

In order to prove the correctness of the conversions of algebraic expressions into periodical sets, it is useful to have a formal result about the transitivity of the *periodically groups into* relation. In addition to transitivity of $\trianglelefteq$, Theorem 1 also says something about period length values.

Theorem 1 *Let G and H be two unbounded granularities such that G is periodic in terms of the bottom granularity (i.e., $\perp \trianglelefteq G$) and H is periodic in terms of G (i.e., $G \trianglelefteq H$). Let P_H^G and N_H^G be the period length and the period label distance of H in terms of granules of G, and N_G the period label distance of G in terms of $\perp$. Then, if $P_H^G = \alpha N_G$ for some positive integer α, then H is periodic in terms of the bottom granularity (i.e., $\perp \trianglelefteq H$) and $P_H = \alpha P_G$.*

Proof. Since by hypothesis $G \trianglelefteq H$ and $P_H^G = \alpha N_G$, $\forall i$ if $H(i) = \bigcup_{r=0}^{n_i} G(i_r)$, then $H(i + N_H^G) = \bigcup_{r=0}^{n_i} G(i_r + \alpha N_G)$. This can be also written as follows:
if

$$H(i) = G(i_0) \cup ... \cup G(i_{n_i}) \tag{1}$$

then $\exists \beta \in \mathbb{N} s.t.$:

$$H(i + N_H^G) = G(i_0 + \alpha N_G) \cup ... \cup G(i_{n_i} + \alpha N_G) \tag{2}$$

Since $\perp \trianglelefteq G$, if

$$G(i_j) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k}) \tag{3}$$

then

$$G(i_j + N_G) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k} + P_G) \tag{4}$$

This can be clearly extended using αN_G instead of N_G.

$$G(i_j + \alpha N_G) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k} + \alpha P_G) \tag{5}$$

Rewriting (1) substituting $G(i_j)$ according to (3) and rewriting (2) substituting $G(i_j + \alpha N_G)$ according to (5), we obtain:
if $H(i) = \underbrace{\perp(i_{0,0}) \cup \ldots \cup \perp(i_{0,\tau_{i_0}})}_{G(i_0)} \cup \ldots \cup \underbrace{\perp(i_{n_i,0}) \cup ... \cup \perp(i_{n_i,\tau_{i_{n_i}}})}_{G(i_{n_i})}$

then $H(i+N_H^G) = \underbrace{\perp(i_{0,0}+\alpha P_G)\cup\ldots\cup\perp(i_{0,\tau_{i_0}}+\alpha P_G)}_{G(i_0+\alpha N_G)}\cup\ldots$
$\cup\underbrace{\perp(i_{n_i,0}+\alpha P_G)\cup\ldots\cup\perp(i_{n_i,\tau_{in_i}}+\alpha P_G)}_{G(i_{n_i}+\alpha N_G)}$
Hence the second condition of Definition 5 is satisfied. The third one is always satisfied for unbounded granularities. The first one is satisfied too; in fact since $G \trianglelefteq H$ with a period label distance of N_H^G, then for each label i of H, $i+N_H^G$ is a label of H. Hence, by definition of periodically-groups-into $\perp \trianglelefteq H$ with $P_H = \alpha P_G$ and $N_H = N_H^G$. □

A.2 Proof of Proposition 1

A.2.1 Part 1

From the definition of the *Group* operation, for all $i \in \mathbb{N}$:

$$G'(i) = \bigcup_{j=(i-1)m+1}^{im} G(j) = G(im-m+1)\cup\ldots\cup G(im) = G(\lambda)\cup\ldots\cup G(\lambda+m-1)$$

with $\lambda = im-m+1$. Furthermore, $\forall k \in \mathbb{N}$:

$$G'(i+k) = \bigcup_{j=(i+k-1)m+1}^{(i+k)m} G(j) = G(im+km-m+1)\cup\ldots\cup G(im+km) =$$

$$= G(\lambda+km)\cup\ldots\cup G(\lambda+km+m-1)$$

Hence,

$$\text{If } G'(i') = \bigcup_{r=0}^{m-1} G(\lambda+r) \text{ then } G'(i'+k) = \bigcup_{r=0}^{m-1} G(\lambda+r+km). \tag{6}$$

This holds for each k. If we use $k = \frac{N_G}{GCM(m,N_G)}$ (note that $k \in \mathbb{N}$), then all the hypotheses of Theorem 1 are satisfied: (i) $\perp \trianglelefteq G$ (by hypothesis); (ii) $G \trianglelefteq G'$ (since $G \trianglelefteq G'$, $\mathcal{L}_{G'} = \mathbb{Z}$, and (6) holds); (iii) $P_{G'}^G = \frac{m\cdot N_G}{GCM(m,N_G)}$ (since we use $k = \frac{N_G}{GCM(m,N_G)}$ and, from (6) we know that $P_{G'}^G = km$). Therefore, by Theorem 1, $\perp \trianglelefteq G'$ with $P_{G'} = \frac{mP_G}{GCM(m,N_G)}$ and $N_{G'} = \frac{N_G}{GCM(m\cdot N_G)}$.

A.2.2 Part 2

By definition of l, we need to show that $G'\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right) = \bigcup_{j=b}^{t} G(j)$ with $b \leq l_G \leq t$.

From the definition of the *Group* operation, $G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i\cdot m} G(i)$; hence:

$$G'\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right) = \bigcup_{j=\left\lfloor\frac{l_G-1}{m}\right\rfloor\cdot m+1}^{\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right)\cdot m} G(j)$$

We prove the thesis showing that (1) $\left\lfloor \frac{l_G - 1}{m} \right\rfloor \cdot m + 1 \leq l_G$ and that (2) $\left(\left\lfloor \frac{l_G - 1}{m} \right\rfloor + 1 \right) \cdot m \geq l_G$.

(1) Since $\left\lfloor \frac{l_G - 1}{m} \right\rfloor \leq \frac{l_G - 1}{m}$, hence $\left\lfloor \frac{l_G - 1}{m} \right\rfloor \cdot m + 1 \leq l_G$

(2) First we prove that $\left\lfloor \frac{l_G - 1}{m} \right\rfloor \geq \frac{l_G}{m} - 1$. Since $\left\lfloor \frac{l_G - 1}{m} \right\rfloor = \frac{l_G - 1 - [(l_G - 1) mod\ m]}{m}$ we have to prove that $\frac{l_G - 1 - [(l_G - 1) mod\ m]}{m} \geq \frac{l_G}{m} - 1$; it is equivalent to the inequality $-[(l_G - 1) mod\ m] \geq -m + 1$ that is true since $(l_G - 1) mod\ m \leq m - 1$. Since $\left\lfloor \frac{l_G - 1}{m} \right\rfloor \geq \frac{l_G}{m} - 1$ it is trivial that $\left(\left\lfloor \frac{l_G - 1}{m} \right\rfloor + 1 \right) \cdot m \geq l_G$.

A.3 Proof of Proposition 2

A.3.1 Part 1

Proof sketch
We show that $G_2 \trianglelefteq G'$ with $P_{G'}^{G_2} = \alpha N_{G_2}$ and then we apply Theorem 1 to obtain the thesis. In particular we use

$$\Delta = lcm \left(N_{G_1}, m, \frac{P_{G_2} \cdot N_{G_1}}{GCD(P_{G_2} \cdot N_{G_1}, P_{G_1})}, \frac{N_{G_2} \cdot m}{GCD(N_{G_2} \cdot m, |k|)} \right)$$

and

$$\alpha = \left(\frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m} \right) \cdot \frac{P_{G_2}}{N_{G_2}}$$

such that, for each i, if $\exists j, k : G'(i) = \bigcup_{r=0}^{k} G_2(j + r)$, then $G'(i + \Delta) = \bigcup_{r=0}^{k} G_2(j + r + \alpha N_{G_2})$.

Given an arbitrary granule $G'(i)$, we show that $G'(i + \Delta)$ is the union of granules that can be obtained by adding αN_{G_2} to the index of each granule of G_2 contained in $G'(i)$. Note that $i + \Delta \in \mathcal{L}_{G'}$ since G' is full-integer labeled. In order to show that this is correct we consider the way granules of G' are constructed by definition of altering-tick. More precisely, we compute the difference between the label $b'_{i+\Delta}$ of the first granule of G_2 included in $G'(i + \Delta)$ and the label b'_i of the first granule of G_2 included in $G'(i)$; we show that this difference is equal to the difference between the label $t'_{i+\Delta}$ of the last granule of G_2 included in $G'(i + \Delta)$ and the label t'_i of the last granule of G_2 included in $G'(i)$. This fact together with the consideration that G_2 is a full-integer labeled granularity, leads to the conclusion that $G'(i)$ and $G'(i + \Delta)$ have the same number of granules. It is then clear that the above computed label differences are also equal to the difference between the label of an arbitrary n-th granule of G_2 included in $G'(i + \Delta)$ and the label of the n-th granule of G_2 included in $G'(i)$. If this difference is $b'_{i+\Delta} - b'_i$, then we have: if $\exists j, k : G'(i) = \bigcup_{r=0}^{k} G_2(j + r)$, then $G'(i + \Delta) = \bigcup_{r=0}^{k} G_2 \left(j + r + \left(b'_{i+\Delta} - b'_i \right) \right)$. By showing that $b'_{i+\Delta} - b'_i$ is a multiple of N_{G_2} the thesis follows.

Proof details

Assume $G_1(i) = \bigcup_{j=b_i}^{t_i} G_2(j)$ and $G_1(i+\Delta) = \bigcup_{j=b_{i+\Delta}}^{t_{i+\Delta}} G_2(j)$. We need to compute $b'_{i+\Delta} - b'_i$. From the definition of the the altering-tick operation:

$$b'_i = \begin{cases} b_i + \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) k & \text{if } i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l, \\ b_i + \left(\left\lfloor \frac{i-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{7}$$

and

$$b'_{i+\Delta} = \begin{cases} b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) k & \text{if } i+\Delta = \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l, \\ b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{8}$$

Note that if $i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l$, then $i + \Delta = \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l$. Indeed, $\left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l = \left(\left\lfloor \frac{i-l}{m} + \frac{\Delta}{m} \right\rfloor\right) m + l$ and, since Δ is a multiple of m, then $\left(\left\lfloor \frac{i-l}{m} + \frac{\Delta}{m} \right\rfloor\right) m + l = \left(\frac{\Delta}{m} + \left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l = \Delta + \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l$.

Hence, to compute $b'_{i+\Delta} - b'_i$ we should consider two cases:

$$b'_{i+\Delta} - b'_i = \begin{cases} b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) k - b_i - \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) k & \text{if } i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l \\ b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor + 1\right) k - b_i - \left(\left\lfloor \frac{i-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{9}$$

In both cases (again considering the fact that Δ is a multiple of m):

$$b'_{i+\Delta} - b'_i = (b_{i+\Delta} - b_i) + \frac{\Delta \cdot k}{m} \tag{10}$$

We are left to compute $b_{i+\Delta} - b_i$, i.e., the distance in terms of granules of G_2, between $G_2(b_i)$ and $G_2(b_{i+\Delta})$. Since, by hypothesis, $G_1(i) = \bigcup_{j=b_i}^{t_i} G_2(j)$ and $G_1(i+\Delta) = \bigcup_{j=b_{i+\Delta}}^{t_{i+\Delta}} G_2(j)$, then the first granule of $\perp$ making $G_2(b_i)$ and the first granule of $\perp$ making $G_1(i)$ is the same granule. The same can be observed for the first granule of $\perp$ making $G_2(b_{i+\Delta})$ and the first granule of $\perp$ making $G_1(i+\Delta)$. More formally:

$$min \lfloor b_i \rfloor^{G_2} = min \lfloor i \rfloor^{G_1}$$

and

$$min \lfloor b_{i+\Delta} \rfloor^{G_2} = min \lfloor i+\Delta \rfloor^{G_1}$$

Hence, we have:

$$min \lfloor b_{i+\Delta} \rfloor^{G_2} - min \lfloor b_i \rfloor^{G_2} = min \lfloor i+\Delta \rfloor^{G_1} - min \lfloor i \rfloor^{G_1} \tag{11}$$

We have shown that the difference between the index of the first granule of $\perp$ making $G_2(b_{i+\Delta})$ and the index of the first granule of $\perp$ making $G_2(b_i)$ is equal to the difference between the index of the first granule of $\perp$ making $G_1(i+\Delta)$ and the index of the first granule of $\perp$ making $G_1(i)$. Then, we need to compute the difference between the index of the first granule of $\perp$ making $G_1(i+\Delta)$ and the index of the first granule of $\perp$ making $G_1(i)$. Since $\perp \trianglelefteq G_1$ and Δ is a multiple of N_{G_1}, for each i, if $\exists j, \tau : G_1(i) = \bigcup_{r=0}^{\tau} \perp(j+r)$, then $G_1(i+\Delta) = \bigcup_{r=0}^{\tau} \bot(j + \frac{\Delta \cdot P_{G_1}}{N_{G_1}})$. Hence, this difference has value $\frac{\Delta \cdot P_{G_1}}{N_{G_1}}$, and for what shown above this is also the value of the difference between the index of the first granule of $\bot$ making $G_2(b_{i+\Delta})$ and the index of the first granule of $\bot$ making $G_2(b_i)$. Then, since $\bot \trianglelefteq G_2$ with period length P_{G_2} and since $\frac{\Delta \cdot P_{G_1}}{N_{G_1}}$ is a multiple of P_{G_2}, we have that, if:

$$\bot(j) \subseteq G_2(i)$$

then:

$$\bot(j + \frac{\Delta \cdot P_{G_1}}{N_{G_1}}) \subseteq G_2(i + \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}})$$

Thus, $b_{i+\Delta} - b_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}}$.

Reconsidering 10:

$$b'_{i+\Delta} - b'_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}.$$

Analogously we can compute $t'_{i+\Delta} - t'_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}$.

Thus, $b'_{i+\Delta} - b'_i = t'_{i+\Delta} - t'_i$; hence $t_{i+\Delta} - b_{i+\Delta} = t_i - b_i$. Since G_2 is a full integer labeled granularity, then $G'(i)$ and $G'(i+\Delta)$ are formed by the same number of granules.

Since we now know $G'(i+\Delta) = \bigcup_{j=b'_{i+\Delta}}^{t'_{i+\Delta}} G_2(j) = \bigcup_{j=b'_i}^{t'_i} G_2(j + (b'_{i+\Delta} - b'_i))$ and $(b'_{i+\Delta} - b'_i)$ is a multiple of N_{G_2}, we have $G_2 \trianglelefteq G'$, $P_{G'}^{G_2} = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}}$ and $\bot \trianglelefteq G_2$. Hence, all the hypothesis of Theorem 1 hold, and its application leads the thesis of this proposition.

A.3.2 Part 2

Since G_2 partitions G' (see table 2.2 of (Bettini et al., 2000)), then (1) $\lceil l_{G_2} \rceil_{G_2}^{G'}$ is always defined and (2) $min(\{n \in \mathbb{N}^+ | \exists i \in \mathcal{L}_{G_2} s.t.\ \bot(n) \subseteq G_2(i)\}) = min(\{m \in \mathbb{N}^+ | \exists j \in \mathcal{L}_{G'} s.t.\ \bot(m) \subseteq G'(j)\})$. Therefore $l_{G'}$ is the label of the granule of G' that covers the granule of G_2 labeled with l_{G_2}; by definition of $\lceil \cdot \rceil$ operation, $l_{G'} = \lceil l_{G_2} \rceil_{G_2}^{G'}$.

A.4 Proof of Proposition 3

A.4.1 Part 2

By definition of the *Shift* operation, $G'(i) = G(i-m)$. Hence $G'(l_G + m) = G(l_G + m - m) = G(l_G)$.

A.5 Proof of Proposition 4

A.5.1 Part 1

The thesis will follow from the application of Theorem 1. Indeed, we know that $\bot \trianglelefteq G_2$ and we show that $G_2 \trianglelefteq G'$ with $P_{G'}^{G_2}$ multiple of N_{G_2}. For this we need to identify Δ and α s.t., for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j \in s(i)} G_2(j)$, then $G'(i+\Delta) = \bigcup_{j \in s(i)} G_2(j + \alpha N_{G_2})$.

Consider an arbitrary $i \in \mathbb{N}$ and $\Delta = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$. By definition of the combining operation, we have $G'(i) = \bigcup_{j \in s(i)} G_2(j)$ and $G'(i+\Delta) = \bigcup_{j \in s(i+\Delta)} G_2(j)$ with

$$s(i) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$$

and

$$s(i+\Delta) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i+\Delta)\}.$$

We now show that $s(i+\Delta)$ is composed by all and only the elements of $s(i)$ when the quantity $\Delta' = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$ is added. For this purpose we need:

$$\forall j \in s(i)\ \exists (j+\Delta') \in s(i+\Delta) \tag{12}$$

and

$$\forall \left(j+\Delta'\right) \in s(i+\Delta)\ \exists j \in s(i) \tag{13}$$

About 12, note that if $j \in s(i)$, then $G_2(j) \subseteq G_1(i)$. Since $\perp \trianglelefteq G_2$, if

$$G_2(j) = \bigcup_{r=0}^{k} \perp(j_r)$$

then

$$G_2\left(j+\Delta'\right) = \bigcup_{r=0}^{k} \perp(j_r + lcm(P_{G_1}, P_{G_2})) \tag{14}$$

Since $G_1(i) \supseteq G_2(j) = \bigcup_{r=0}^{k} \perp(j_r)$, and since $\perp \trianglelefteq G_1$, then

$$G_1(j+\Delta) \supseteq \bigcup_{r=0}^{k} \perp(j_r + lcm(P_{G_1}, P_{G_2})) \tag{15}$$

From 14 and 15 we derive $G_1(i+\Delta) \supseteq G_2(j+\Delta')$, and hence $(j+\Delta') \in s(i+\Delta)$. Analogously can be proved the validity of 13; Hence, for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j \in s(i)} G_2(j)$, then $G'(i+\Delta) = \bigcup_{j \in s(i)} G_2(j+\Delta')$. Hence, considering the fact that $G_2 \trianglelefteq G'$, we can conclude $G_2 \trianglelefteq G'$. Finally, since $P_{G'}^{G_2}$ is a multiple of N_{G_2}, by Theorem 1 we obtain the thesis.

A.5.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \widetilde{s}(i) \neq \emptyset\}$$

where $\forall i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}\ \widetilde{s}(i) = \{j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$;

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that: (1) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that exists $k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$ and since G' is derived by the *Combine* operation, then $\exists q \in \mathcal{L}_{G_2} | G_2(q) \subseteq G_1(k)$. By definition of the *Combine* operation $G'(k) = \bigcup_{j \in s(k)} G_2(j)$; since $q \in s(k)$, then $G_2(q) \subseteq G'(k)$. Hence (a) $\exists q \in \mathcal{L}_{G_2} | G_2(q) \subseteq G'(k)$.

Moreover, since $k \notin \widetilde{\mathcal{L}}_{G'}$, then $\widetilde{s}(k) = \emptyset$; therefore $\nexists \mathbf{j} \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(k)$. By definition of the *Combine* operation it is easily seen that $G' \preceq G_1$. Using this and the previous formula, we derive that (b) $\nexists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G'(k)$.

From (a) and (b) it follows that $\exists q \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(q) \subseteq G'(k)$. We show that this leads to a contradiction.

Since $q \notin \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ and labels of $\hat{\mathcal{L}}_{G_2}^{P_{G'}}$ are contiguous (i.e., $\nexists i \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ s.t. $min(\hat{\mathcal{L}}_{G_2}^{P_{G'}}) < i < max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$), then $q < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ or $q > max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$. We consider the first case, the proof for the second is analogous.

If $q < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ then $max(\lfloor q \rfloor^{G_2}) < 1$ (otherwise $q \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$).

Let be $\alpha = min(\lfloor min(\hat{\mathcal{L}}_{G'}) \rfloor^{G'})$. Since $k \in \hat{\mathcal{L}}_{G'}$, then $\alpha \leq \lfloor k \rfloor^{G'}$.

If $\alpha \geq 1$, then $G'(k) \cap G_2(q) = \emptyset$ contradicting $G'(k) \supseteq G_2(q)$.

If $\alpha < 1$, then $G'(l_{G'}) \supseteq \perp(0)$ and we show that $l_{G'} \in \widetilde{\mathcal{L}}_{G'}$. Indeed, by definition of *Combine*, $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G'(L_{G'})$. Since $G' \preceq G_1$ we also have $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(L_{G'})$; hence $j \in \widetilde{s}(l_{G'})$ and then $l_{G'} \in \widetilde{\mathcal{L}}_{G'}$.

Since $0 \in G'(l_{G'})$ and $max(\lfloor q \rfloor^{G_2}) \leq 0$, then $max(\lfloor q \rfloor^{G_2}) < \alpha$ (otherwise $G_2(q) \subseteq G'(l_{G'})$). Therefore, since $min(\lfloor k \rfloor^{G'}) \geq \alpha$, then $\lfloor q \rfloor^{G_2} \cap \lfloor l_{G'} \rfloor^{G'} = \emptyset$, in contradiction with $G_2(q) \subseteq G'(k)$.

(2) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$, by definition of $\widetilde{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ and $\widetilde{s}(k) \neq \emptyset$; Therefore, by definition of $\widetilde{s}$, $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(k)$.

Since $j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$, by definition of $\hat{\mathcal{L}}$, $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_2} = j$. Since $G_2(j) \subseteq G_1(k)$, then $\lceil h \rceil^{G_1} = k$. By definition of the *combine* operation, $\lceil h \rceil^{G'} = k$. Moreover, since $0 < h \leq P_{G'}$, by definition of $\hat{\mathcal{L}}$, $\lceil h \rceil^{G'} = k \in \hat{\mathcal{L}}_{G'}$, contradicting the hypothesis.

A.6 Proof of Proposition 5

A.6.1 Part 1

The thesis will follow from the application of Theorem 1. Indeed, we show that $G_1 \trianglelefteq G'$ with $P_{G'}^{G_1}$ multiple of N_{G_1}. For this we need to identify Δ and α s.t., for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j \in s(i)} G_1(j)$, then $G'(i+\Delta) = \bigcup_{j \in s(i)} G_1(j+\alpha N_{G_1})$. Let $\Delta = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$. By definition of anchored grouping, $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ and $G'(i+\Delta) = \bigcup_{j=i+\Delta}^{(i+\Delta)'-1} G_1(j)$ where i' is the first label of G_2 after i and $(i+\Delta)'$ is the first label of G_2 after $i+\Delta$. By periodicity of G_2, (and since Δ is a multiple of N_{G_2}) the difference between the label of the granule following $G_2(i+\Delta)$ and the label of the granule following $G_2(i)$ is Δ. More formally, $(i+\Delta)' - i' = \Delta$, hence $(i+\Delta)' = i' + \Delta$. Then, for each i, if $G'(i) = \bigcup_{j=i}^{k} G_1(j)$, then $G'(i+\Delta) = \bigcup_{j=i+\Delta}^{i'+\Delta-1} G_1(j) = \bigcup_{j=i}^{i'-1} G_1(j+\Delta)$. By this result and considering $G_1 \trianglelefteq G'$, we conclude $G_1 \trianglelefteq G'$ with $P_{G'}^{G_1} = \Delta$. Note that by Proposition 9, $N_{G_1} = \frac{P_{G_1} \cdot N_{G_2}}{P_{G_2}}$, hence $P_{G'}^{G_1}$ is a multiple of Δ. Then, by Theorem 1, we have the thesis.

A.6.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \begin{cases} \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{if } l_{G_2} = l_{G_1}, \\ \{l'_{G_2}\} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{otherwise,} \end{cases}$$

where l'_{G_2} is the greatest among the labels of $\mathcal{L}_{G_2}$ that are smaller than l_{G_2}. We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\hat{\mathcal{L}}_{G'} \subseteq \widetilde{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Then, since $k \in \widetilde{\mathcal{L}}_{G'}$, then $k \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}$ or $k = l'_{G_2}$.

If $k \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}$, then, by definition of $\hat{\mathcal{L}}^{P_{G'}}_{G_2}$, $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_2} = k$. By definition of *Anchored-group*, $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ where k' is the first label of G_2 after k. Therefore $G'(k) \supseteq G_1(k)$. Since G_2 is a labeled aligned subgranularity of G_1 and since $k \in \mathcal{L}_{G_2}$, then $k \in \mathcal{L}_{G_1}$ and $G_1(k) = G_2(k)$. Hence $G'(k) \supseteq G_2(k)$. It follows that $\lceil h \rceil^{G'} = k$ and therefore, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

If $k = l'_{G_2}$, then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $l_{G_2} \neq l_{G_1}$. Therefore, since G_2 is a labeled aligned subgranularity of G_1 $l'_{G_2} < l_{G_1} < l_{G_2}$; then $\exists h$ with $0 < h < min(\lfloor l_{G_2} \rfloor^{G_2})$ s.t. $\lceil h \rceil^{G_1} = l_{G_1}$. Since, by definition of *Anchored-group*, $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$ and since $l'_{G_2} < l_{G_1} < l_{G_2}$, then $G'(l'_{G_2}) \supseteq G_1(l_{G_1})$. Hence $\lceil h \rceil^{G'} = l'_{G_2}$ and therefore, by definition of $\hat{\mathcal{L}}$, $l'_{G_2} = k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. If $k \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}$ then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $k \in \widetilde{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

If $k \notin \hat{\mathcal{L}}^{P_{G'}}_{G_2}$, since $\nexists q \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}^{P_{G'}}_{G_2}$ s.t. $min(\hat{\mathcal{L}}^{P_{G'}}_{G_2}) \leq q \leq max(\hat{\mathcal{L}}^{P_{G'}}_{G_2})$, then $k > max(\hat{\mathcal{L}}^{P_{G'}}_{G_2})$ or $k < min(\hat{\mathcal{L}}^{P_{G'}}_{G_2})$.

If $k > max(\hat{\mathcal{L}}^{P_{G'}}_{G_2})$ then, by definition of $\hat{\mathcal{L}}$, $min(\lfloor k \rfloor^{G_2}) > P_{G'}$. Since G_2 is a labeled aligned subgranularity of G_1 then $G_2(k) = G_1(k)$ and hence $min(\lfloor k \rfloor^{G_1}) > P_{G'}$. Since $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ then $min(\lfloor k \rfloor^{G'}) > P_{G'}$ in contrast with the hypothesis $k \in \hat{\mathcal{L}}_{G'}$.

If $k < min(\hat{\mathcal{L}}^{P_{G'}}_{G_2})$ then, by definition of l'_{G_2}, $k < l'_{G_2}$ or $k = l'_{G_2}$.

If $k < l'_{G_2}$ then, let k' be the next label of G_2 after k. Since $k < l'_{G_2}$ then, by definition l'_{G_2}, $k' \leq l'_{G_2}$. By definition of l'_{G_2} then $max(\lfloor l'_{G_2} \rfloor^{G_2}) \leq 0$. Since G_2 is a labeled aligned subgranularity of G_1 then $G_1(l'_{G_2}) = G_2(l'_{G_2})$; therefore $max(\lfloor l'_{G_2} \rfloor^{G_1}) \leq 0$. Since $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ and $k' \leq l'_{G_2}$, follows that $max(\lfloor k \rfloor^{G'}) \leq 0$ in contrast with the hypothesis $k \in \hat{\mathcal{L}}_{G'}$.

Finally if $k = l'_{G_2}$ then $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$. Since $k = l'_{G_2} \in \hat{\mathcal{L}}_{G'}$ then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = l'_{G_2}$. Since G' is the composition of granules of G_1, $\lceil h \rceil^{G_1}$ is defined. Let $q = \lceil h \rceil^{G_1}$. By definition of $\hat{\mathcal{L}}$, $q \in \hat{\mathcal{L}}^{P_{G'}}_{G_1}$ and therefore $q \geq l_{G_1}$. Since, by definition of *Anchored-group*, G' is the composition of granules of G_1 and since $\lceil h \rceil^{G'} = l'_{G_2}$ and $\lceil h \rceil^{G_1} = q$, then $G_1(q) \subseteq G'(l'_{G_2})$. Therefore since $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$ then $q < l_{G_2}$. It follows that $l_{G_1} \leq q < l_{G_2}$ and hence $l_{G_1} \neq l_{G_2}$. By definition of $\widetilde{\mathcal{L}}_{G'}$, $l'_{G_2} = k \in \widetilde{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

A.7 Selecting operations

The selecting operations have a common part in the proof for the computation of the period length and the period label distance.

Let be $\Gamma = \frac{lcm(P_{G_1}, P_{G_2})N_{G_1}}{P_{G_1}}$. The proof is divided into two steps: first we show that for each select operation if $i \in \mathcal{L}_{G'}$ then $i + \Gamma \in \mathcal{L}_{G'}$ (details for *Select-down*, *Select-up* and *Select-by-intersect* operations can be found below). The second step is the application of Theorem 1. Indeed, for each *Select* operation, the following holds: $\forall i \in \mathcal{L}_{G'}\ G'(i) = G_1(i)$; this implies $G_1 \trianglelefteq G'$. From step 1 follows that $i + \Gamma \in \mathcal{L}_{G'}$, hence $G'(i + \Gamma) = G_1(i + \Gamma)$. By this result and considering $G_1 \trianglelefteq G'$, we conclude that $G_1 \bar{\trianglelefteq} G'$ with $P_{G'}^{G_1} = \Gamma$ which is a multiple of N_{G_1} by definition. Then, by Theorem 1 we have the thesis.

A.8 Proof of Proposition 6

A.8.1 Part 1

See Section A.7.

We prove that if $\lambda \in \mathcal{L}_{G'}$ then $\lambda' = \lambda + \Gamma \in \mathcal{L}_{G'}$.

By definition of the `select-down` operation, if $\lambda \in \mathcal{L}_{G'}$ then $\exists i \in \mathcal{L}_{G_2}$ s.t. $\lambda \in \Delta_k^l(S(i))$ where $S(i)$ is an ordered set defined as follows: $S(i) = \{j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i)\}$. In order to prove the thesis we need to show that $\exists i' \in \mathcal{L}_{G_2} | \lambda' \in \Delta_k^l(S(i'))$. Consider $i' = i + \frac{lcm(P_{G_1} P_{G_2})N_{G_2}}{P_{G_2}}$ we will note that $i' \in \mathcal{L}_{G_2}$ (this is trivially derived from the periodicity of G_2). To prove that $\lambda' \in \Delta_k^l(S(i'))$ we show that $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements.

Indeed note that from periodicity of G_1, $\forall j \in S(i)$ if:

$$G_1(j) = \bigcup_{r=0}^{\tau_j} \perp(j_r) \tag{16}$$

then:

$$G_1(j') = \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1} P_{G_2})) \tag{17}$$

Since $j \in S(i)$, $G_1(j) \subseteq G_2(i)$ then, from (16), $G_2(i) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r)$. Moreover, from periodicity of G_2:

$$G_2(i') \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1} P_{G_2})) \tag{18}$$

Since (17) and (18), $G_2(i') \supseteq G_1(j')$; hence $\forall j \in S(i), j' = (j + \Gamma) \in S(i')$. Analogously we can prove that $\forall j' \in S(i'), j = (j' - \Gamma) \in S(i)$.

Thus $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements; therefore if $j \in S(i)$ has position n in $S(i)$, so $j' \in S(i')$ has position n in $S(i')$. Hence it is trivial that if λ has position between k and $k + l - 1$ in $S(i)$, then λ' has position between k and $k + l - 1$ in $S(i')$. Hence if $\lambda \in \mathcal{L}_{G'}$, then $\lambda' \in \mathcal{L}_{G'}$.

A.8.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}} \left\{ a \in A(i) | a \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} \right\};$$

where $\forall i \in \mathcal{L}_{G_2}$:

$$A(i) = \Delta_k^l \left(\{ j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i) \} \right).$$

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1)Suppose by contradiction that $\exists q \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. By definition of $\widetilde{\mathcal{L}}_{G'}$, $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$; therefore $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_1} = q$. Moreover, by definition of $\widetilde{\mathcal{L}}_{G'}$ and by definition of *Select-down*, $\widetilde{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$ hence $q \in \mathcal{L}_{G'}$. Since, by definition of *Select-down* $G'(q) = G_1(q)$, then $\lceil h \rceil^{G'} = q$; hence, by definition of $\hat{\mathcal{L}}$, $q \in \hat{\mathcal{L}}_{G'}$ in contradiction with hypothesis.

(2)Suppose by contradiction that $\exists q \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $q \in \hat{\mathcal{L}}_{G'}$ then, by definition of *Select-down*

$$\exists i \in \mathcal{L}_{G_2}\, s.t.\, q \in \Delta_k^l \left(\{ j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i) \} \right)$$

therefore, by definition of $A(i)$, $q \in A(i)$.

Since $q \in \hat{\mathcal{L}}_{G'}$ then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = q$. By definition of *Select-down*, $G'(q) = G_1(q)$, then $\lceil h \rceil^{G_1} = q$ and therefore $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$. Moreover, since $G_1(q) \subseteq G_2(i)$, then $\lceil h \rceil^{G_2} = i$ and therefore $i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$. Since $q \in A(i)$, $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ and $i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $q \in \widetilde{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

A.9 Proof of Proposition 7

A.9.1 Part 1

See Section A.7. We prove that if $i \in \mathcal{L}_{G'}$ then $i + \Gamma \in \mathcal{L}_{G'}$. From the periodicity of G_1, $i + \Gamma \in \mathcal{L}_{G_1}$ (this is trivially derived from the periodicity of G_1). Hence we only need to show that $\exists j' \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i + \Gamma)$. Since $i \in \mathcal{L}_{G'}$ then $\exists j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)$.

From the periodicity of G_2, if:

$$G_2(j) = \bigcup_{r=0}^{\tau_j} \perp(j_r) \tag{19}$$

then:

$$G_2\left(j + \frac{lcm(P_{G_1} P_{G_2}) N_{G_2}}{P_{G_2}}\right) = \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1} P_{G_2})) \tag{20}$$

Moreover, from the (19) and since $G_1(i) \supseteq G_2(j)$:

$$G_1(i) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r)$$

From the periodicity of G_1:

$$G_1(i+\Gamma) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1}P_{G_2})) \tag{21}$$

From (20) and (21) follows that $G_1(i+\Gamma) \supseteq G_2\left(j + \frac{lcm(P_{G_1}P_{G_2})N_{G_2}}{P_{G_2}}\right)$, that is the thesis.

A.9.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \exists j \in \mathcal{L}_{G_2}\, s.t.\, \emptyset \neq G_2(j) \subseteq G_1(i)\};$$

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G_2}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$, then $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$; therefore $\exists h$ with $0 < h \leq P_{G'}$ s. t. $\lceil h \rceil^{G_1} = k$. Moreover, by definition of $\widetilde{\mathcal{L}}_{G'}$ and by definition of *Select-down*, $\widetilde{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$ hence $q \in \mathcal{L}_{G'}$. Since, by definition of *Select-up*, $G'(k) = G_1(k)$, then $\lceil h \rceil^{G'} = k$. Hence, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$, then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Since, by definition of *Select-up*, $G'(k) = G_1(k)$, then $\lceil h \rceil^{G_1} = k$; Therefore, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$. Moreover, since $k \in \hat{\mathcal{L}}_{G'}$ and $\hat{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$, by definition of the *Select-up* operation, then $\exists j \in \mathcal{L}_{G_2}$ s.t. $\emptyset \neq G_2(j) \subseteq G_1(k)$. Hence by definition of $\widetilde{\mathcal{L}}_{G'}$, $k \in \widetilde{\mathcal{L}}_{G'}$, in contradiction with hypothesis.

A.10 Proof of Proposition 8

A.10.1 Part 1

See Section A.7. We prove that if $\lambda \in \mathcal{L}_{G'}$, then $\lambda' = \lambda + \Gamma \in \mathcal{L}_{G'}$.

By definition of the *select-by-intersect* operation, if $\lambda \in \mathcal{L}_{G'}$, then $\exists i \in \mathcal{L}_{G_2} : \lambda \in \Delta_k^l(S(i))$ where $S(i)$ is an ordered set defined as follows: $S(i) = \{j \in \mathcal{L}_{G_1} | G_1(j) \cap G_2(i) \neq \emptyset\}$. In order to prove the thesis we need to show that $\exists i' \in \mathcal{L}_{G_2} : \lambda' \in \Delta_k^l(S(i'))$. Consider $i' = i + \frac{lcm(P_{G_1}P_{G_2})N_{G_2}}{P_{G_2}}$ note that $i' \in \mathcal{L}_{G_2}$ (this is trivially derived from the periodicity of G_2). To prove that $\lambda' \in \Delta_k^l(S(i'))$ we show that $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements.

Indeed note that $\forall j$ if $j \in S(i)$, then $G_1(j) \cap G_2(i) \neq \emptyset$. Hence $\exists l \in \mathbb{Z} : \perp(l) \subseteq G_1(j)$ and $\perp(l) \subseteq G_2(i)$. From the periodicity of G_1, $G_1(j+\Gamma) \supseteq \perp(l + lcm(P_{G_1}P_{G_2}))$. From the periodicity of G_2, $G_2(i') \supseteq \perp(l + lcm(P_{G_1}P_{G_2}))$. So $G_1(j+\Gamma) \cap G_2(i') \neq \emptyset$, therefore $\forall j \in S(i), (j+\Gamma) \in S(i')$.

Analogously we can prove that $\forall j' \in S(i'), (j' - \Gamma) \in S(i)$. Hence $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements. Therefore, if $j \in S(i)$ has position n in $S(i)$, then $j + \Gamma \in S(i')$ has position n in $S(i')$; hence if j has position between k and $k+l-1$ in $S(i)$, then also $j+\Gamma$ has position between k and $k+l-1$ in $S(i')$ and so $j + \Gamma \in \mathcal{L}_{G'}$.

A.10.2 Part 2

The proof is analogous to the ones of Proposition 6.

A.11 Set Operations

A.11.1 Proof of Proposition 9

Given the periodical granularities H and G with G label aligned subgranularity of H, we prove that $\frac{N_G}{P_G} = \frac{N_H}{P_H}$. The thesis is proved by considering the common period length of H and G i.e. $P_c = lcm(P_G, P_H)$.

Let N'_G be the difference between the label of the i^{th} granule of one period of G and the label of the i^{th} granule of the next period, considering P_c as the period length of G. Analogously N'_H is defined.

By periodicity of G, if $G(i) = \bigcup_{r=0}^{k} \perp(i_r)$ then $G(i + N'_G) = \bigcup_{r=0}^{k} \perp(i_r + P_c)$; since G is an aligned subranularity of H, $\forall i \in \mathcal{L}_H$ $H(i) = G(i) = \bigcup_{r=0}^{k} \perp(i_j)$ and, since H is periodic, $H(i+N'_H) = \bigcup_{r=0}^{k} \perp(i_j + P_c)$; from which we can easily derive that $i + N'_G = i + N'_H$, hence $N'_G = N'_H$.

From the definition of P_c, $\exists \alpha, \beta \in \mathbb{N}$ s. t. $\alpha P_H = \beta P_G$. Moreover, since $N'_H = N'_G$, then $\alpha N_H = \beta N_G$. Therefore $\frac{P_H}{N_H} = \frac{P_G}{N_G}$.

A.11.2 Property used in the proofs for set operations

Let Γ_1 be $\frac{lcm(P_{G_1},P_{G_2})N_{G_1}}{P_{G_1}}$ and Γ_2 be $\frac{lcm(P_{G_1},P_{G_2})N_{G_2}}{P_{G_2}}$. Since G_1 and G_2 are aligned subgranularity of a certain granularity H, from Proposition 9 we can easily derive that $\Gamma_1 = \Gamma_2$.

A.12 Proof of Proposition 10

A.12.1 Part 1

Union. Let Γ_1 be $\frac{lcm(P_{G_1},P_{G_2})N_{G_1}}{P_{G_1}}$ and Γ_2 be $\frac{lcm(P_{G_1},P_{G_2})N_{G_2}}{P_{G_2}}$. The thesis will be proved by showing that $\forall i \in \mathcal{L}_{G'}$ if, $G'(i) = \bigcup_{r=0}^{k} \perp(i_r)$, then $G'(i+\Delta) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$ with $\Delta = \Gamma_1 = \Gamma_2$. Since $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cup \mathcal{L}_{G_2}$, two cases will be considered:

- $\forall i \in \mathcal{L}_{G_1}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_1, $G_1(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence $G'(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$.
- $\forall i \in \mathcal{L}_{G_2} - \mathcal{L}_{G_1}$ $G'(i) = G_2(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_2, $G_2(i+\Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence $G'(i + \Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$.

Since $\Gamma_1 = \Gamma_2$, then $\forall i \in \mathcal{L}_{G'}$ if $G'(i) = \bigcup_{r=0}^{k} \perp(i_r)$, then $G'(i + \Gamma_1) = G'(i + \Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$. Hence, by definition of $\trianglelefteq$, we have the thesis.

Intersect. $\forall i \in \mathcal{L}_{G'} = \mathcal{L}_{G_1} \cap \mathcal{L}_{G_2}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_1 and G_2, $i + \Gamma_1 \in \mathcal{L}_{G_1}$ e $i + \Gamma_2 \in \mathcal{L}_{G_2}$; since $\Gamma_1 = \Gamma_2$, then $i + \Gamma_1 \in \mathcal{L}_{G'}$. Moreover $G'(i + \Gamma_1) = G_1(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence, by the definition of $\trianglelefteq$, we have the thesis.

Difference. $\forall i \in \mathcal{L}_{G'} = \mathcal{L}_{G_1} - \mathcal{L}_{G_2}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. Since $i \in \mathcal{L}_{G_1}$ from the periodicity of G_1 $i + \Gamma_1 \in \mathcal{L}_{G_1}$. Since $i \notin \mathcal{L}_{G_2}$, from the periodicity of G_2, $i + \Gamma_2 \notin \mathcal{L}_{G_2}$ (if it would exists $i + \Gamma_2 \in \mathcal{L}_{G_2}$, from periodicity of G_2 would exists $i \in \mathcal{L}_{G_2}$ that is not possible for hypothesis). Hence $i + \Gamma_1 \in \mathcal{L}_{G'}$. Moreover $G'(i + \Gamma_1) = G_1(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence, by the definition of $\trianglelefteq$, we have the thesis.

A.12.2 Part 2

Let $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G_1}^{P_{G'}} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}$.

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$ then $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ or $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$. Suppose that $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ (the proof is analogous if $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$). Since $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$, then $\exists\ 0 < h < P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Since, by definition of the *Union* operation $G'(k) = G_1(k)$, then $\lceil h \rceil^{G'} = k$. Hence, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$, then, by definition of $\hat{\mathcal{L}}$, $\exists\ 0 < h < P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Moreover, by definition of the *Union* operation, $k \in \mathcal{L}_{G_1}$ or $k \in \mathcal{L}_{G_2}$. Suppose that $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ (the proof is analogous if $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$). By definition of the *Union* operation, $G'(k) = G_1(k)$ therefore $\lceil h \rceil^{G_1} = k$ and so, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$. Hence, by definition of $\widetilde{\mathcal{L}}$, $k \in \widetilde{\mathcal{L}}_{G'}$ in contradiction with the hypothesis.

References

Bettini, C., & Mascetti, S. (2005). An efficient algorithm for minimizing time granularity periodical representations. In Proc. of the 12th International Symposium on Temporal Representation and Reasoning (TIME), pp. 20–25. IEEE Computer Society.

Bettini, C., Mascetti, S., & Pupillo, V. (2005). A system prototype for solving multi-granularity temporal csp. In Recent Advances in Constraints, Revised selected papers from the Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)*, volume 3419 of* Lecture Notes in Computer Science, pp. 142–156. Springer.

Bettini, C., Mascetti, S., & Wang., X. S. (2004). Mapping calendar expressions into periodical granularities. In Proc. of the 11th International Symposium on Temporal Representation and Reasoning (TIME), pp. 96–102. IEEE Computer Society.

Bettini, C., & Sibi, R. D. (2000). Symbolic representation of user-defined time granularities. *Annals of Mathematics and Artificial Intelligence*, *30*(1-4), 53–92.

Bettini, C., Wang, X. S., & Jajodia, S. (2000). *Time Granularities in Databases, Data Mining, and Temporal Reasoning.* Springer.

Bettini, C., Wang, X. S., & Jajodia, S. (2002a). Solving multi-granularity temporal constraint networks. *Artificial Intelligence*, *140*(1/2), 107–152.

Bettini, C., Wang, X. S., & Jajodia, S. (2002b). Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, *11*(3), 269–306.

Bresolin, D., Montanari, A., & Puppis, G. (2004). Time granularities and ultimately periodic automata. In Proc. of the 9th European Conference on Logics in Artificial Intelligence (JELIA) *volume 3229 of* Lecture Notes in Computer Science, pp. 513–525. Springer.

Chandra, R., Segev, A., & Stonebraker, M. (1994). Implementing calendars and temporal rules in next generation databases. In Proc. of the 10th International Conference on Data Engineering (ICDE), pp. 264–273. IEEE Computer Society.

Combi, C., Franceschet, M., & Peron, A. (2004). Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, *14*(1), 51–77.

Combi, C., & Pozzi, G. (2003). Temporal conceptual modelling of workflows. In Proc. of the 22nd International Conference on Conceptual Modeling (ER) *volume 2813 of* Lecture Notes in Computer Science, pp. 59–76. Springer.

Cukierman, D., & Delgrande, J. P. (1998). Expressing time intervals and repetition within a formalization of calendars. *Computational Intelligence*, *14*, 563–597.

Dal Lago, U., & Montanari, A. (2001). Calendars, time granularities, and automata. In Proc. of the 7th International Symposium on Spatial and Temporal Databases (SSTD), *volume 2121 of* Lecture Notes in Computer Science, pp. 279–298. Springer.

Dal Lago, U., Montanari, A., & Puppis, G. (2003). Towards compact and tractable automaton-based representations of time granularities. In Proc. of the 8th Italian Conference on Theoretical Computer Science (ICTCS), *volume 2841 of* Lecture Notes in Computer Science, pp. 72–85. Springer.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, *49*(1-3), 61–95.

Kabanza, F., Stevenne, J. M., & Wolper, P. (1990). Handling infinite temporal data. In Proc. of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), pp. 392–403. ACM Press.

Koomen, J. (1991). Reasoning about recurrence. *International Journal of Intelligent Systems*, *6*, 461–496.

Ladkin, P. B. (1986). Primitives and units for time specification. In Proc. of the 5th National Conference on Artificial Intelligence (AAAI), pp. 353–359. Morgan Kaufmann.

Leban, B., McDonald, D., & Forster, D. (1986). A representation for collections of temporal intervals. In Proc. of the 5th National Conference on Artificial Intelligence (AAAI), pp. 367–371. Morgan Kaufmann.

Li, Y., Ning, P., Wang, X. S., & Jajodia, S. (2001). Discovering calendar-based temporal association rules. In Proc. of the 8th International Symposium on Temporal Representation and Reasoning (TIME), pp. 111–118. IEEE Computer Society.

Montanari, A. (1996). *Metric and Layered Temporal Logic for Time Granularity.* Ph.D. thesis, ILLC Dissertation Series 1996-02, University of Amsterdam.

Morris, R., Shoaff, W., & Khatib, L. (1996). Domain-independent temporal reasoning with recurring events. *Computational Intelligence*, *12*, 450–477.

Niezette, M., & Stevenne, J. M. (1992). An efficient symbolic representation of periodic time. In Proc. of the first International Conference on Information and Knowledge Management (CIKM) *volume 725 of* Lecture Notes in Computer Science, pp. 161–168. Springer.

Ning, P., Wang, X. S., & Jajodia, S. (2002). An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, *36*(1-2), 5–38.

Puppis, G. (2006). *Automata for Branching and Layered Temporal Structures.* Ph.D. thesis, Università degli Studi di Udine.

Terenziani, P. (2003). Symbolic user-defined periodicity in temporal relational databases. *IEEE Transactions of Knowledge and Data Engineering*, *15*(2), 489–509.

Tuzhilin, A., & Clifford, J. (1995). On periodicity in temporal databases. *Information Systems*, *20*(8), 619–639.

Urgun, B., Dyreson, C. E., Snodgrass, R. T., Miller, J. K., Soo, M. D., Kline, N., & Jensen, C. S. (2007). Integrating multiple calendars using TauZaman. *Software-Practice Experience*, *to appear*.

Wijsen, J. (2000). A string-based model for infinite granularities. In Spatial and Temporal Granularity: Papers from the AAAI Workshop. *Technical Report WS-00-08*, pp. 9–16. AAAI Press.

Journal of Artificial Intelligence Research 28 (2007) 299-348 Submitted 07/06; published 03/07

Supporting Temporal Reasoning by Mapping Calendar Expressions to Minimal Periodic Sets

Claudio Bettini BETTINI@DICO.UNIMI.IT
Sergio Mascetti MASCETTI@DICO.UNIMI.IT
Dipartimento di Informatica e Comunicazione, Università di Milano
Via Comelico, 39, 20135, Milan, Italy

X. Sean Wang SEAN.WANG@UVM.EDU
Department of Computer Science, University of Vermont
33 Colchester Avenue, Burlington, VT, 05405 USA

Abstract

In the recent years several research efforts have focused on the concept of time granularity and its applications. A first stream of research investigated the mathematical models behind the notion of granularity and the algorithms to manage temporal data based on those models. A second stream of research investigated symbolic formalisms providing a set of algebraic operators to define granularities in a compact and compositional way. However, only very limited manipulation algorithms have been proposed to operate directly on the algebraic representation making it unsuitable to use the symbolic formalisms in applications that need manipulation of granularities.

This paper aims at filling the gap between the results from these two streams of research, by providing an efficient conversion from the algebraic representation to the equivalent low-level representation based on the mathematical models. In addition, the conversion returns a minimal representation in terms of period length. Our results have a major practical impact: users can more easily define arbitrary granularities in terms of algebraic operators, and then access granularity reasoning and other services operating efficiently on the equivalent, minimal low-level representation. As an example, we illustrate the application to temporal constraint reasoning with multiple granularities.

From a technical point of view, we propose an hybrid algorithm that interleaves the conversion of calendar subexpressions into periodical sets with the minimization of the period length. The algorithm returns set-based granularity representations having minimal period length, which is the most relevant parameter for the performance of the considered reasoning services. Extensive experimental work supports the techniques used in the algorithm, and shows the efficiency and effectiveness of the algorithm.

1. Introduction

According to a 2006 research by Oxford University Press, the word *time* has been found to be the most common noun in the English language, considering diverse sources on the Internet including newspapers, journals, fictions and weblogs. What is somehow surprising is that among the 25 most common nouns we find time granularities like *day*, *week*, *month* and *year*. We are pretty sure that many other time granularities like *business day*, *quarter*, *semester*, etc. would be found to be quite frequently used in natural languages. However, the way computer applications deal with these concepts is still very naive and mostly hidden in program code and/or based on limited and sometimes imprecise calendar support.

Temporal representation and reasoning has been for a long time an AI research topic aimed at providing a formal framework for common sense reasoning, natural language understanding, planning, diagnosis and many other complex tasks involving time data management. Despite the many relevant contributions, time granularity representation and reasoning support has very often been ignored or over-simplified. In the very active area of temporal constraint satisfaction, most proposals implicitly assumed that adding support for granularity was a trivial extension. Only quite recently it was recognized that this is not the case and specific techniques were proposed (?). Even the intuitively simple task of deciding whether a specific instant is part of a time granularity can be tricky when arbitrary user-defined granularities like e.g., *banking days*, or *academic semesters* are considered.

Granularities and periodic patterns in terms of granularities are playing a role even in emerging application areas like inter-organizational workflows and personal information management (PIM). For example, inter-organizational workflows need to model and monitor constraints like: *Event2 should occur no later than two business days after the occurrence of Event1.* In the context of PIM, current calendar applications, even on mobile devices, allow the user to specify quite involved periodical patterns for the recurrence of events. For example, it is possible to schedule an event every last Saturday of every two months. The complexity of the supported patterns has been increasing in the last years, and the current simple interfaces are showing their limits. They are essentially based on a combination of recurrences based on one or two granularities taken from a fixed set (days, weeks, months, and years). We foresee the possibility for significant extensions of these applications by specifying recurrences over user-defined granularities. For example, the user may define (or upload from a granularity library) the granularity corresponding to the academic semester of the school he is teaching at, and set the date of the finals as the last Monday of each semester. A bank may want to define its *banking days* granularity and some of the bank policies may then be formalized as recurrences in terms of that granularity. Automatically generated appointments from these policies may appear on the devices of bank employees involved in specific procedures. We also foresee the need to show a user preferred view of the calendar. With current standard applications the user has a choice between a business-day limited view and a complete view, but why not enabling a view based on the users's *consulting-days*, for example? A new perspective in the use of mobile devices may also result from considering the time span in which activities are supposed to be executed (expressed in arbitrary granularities), and having software agents on board to alert about constraints that may be violated, even based on contextual information like the user location or traffic conditions. This scenario highlights three main requirements: a) a sufficiently expressive formal model for time granularity, b) a convenient way to define new time granularities, and c) efficient reasoning tools over time granularities.

Consider a). In the last decade significant efforts have been made to provide formal models for the notion of time granularity and to devise algorithms to manage temporal data based on those models. In addition to *logical* approaches (?, ?), a framework based on periodic-set representations has been extensively studied (?), and more recently an approach based on strings and automata was introduced (?, ?). We are mostly interested in the last two approaches because they support the effective computation of basic operations on time granularities. In both cases the representation of granularities can be considered as a *low-level* one, with a rather involved specification in terms of the instants of the time domain.

Consider requirement b) above. Users may have a hard time in defining granularities in formalisms based on low-level representations, and to interpret the output of operations. It is clearly unreasonable to ask users to specify granularities by linear equations or other mathematical formalisms that operate directly in terms of instants or of granules of a fixed time granularity. Hence, a second stream of research investigated more *high-level* symbolic formalisms providing a set of algebraic operators to define granularities in a compact and compositional way. The efforts on this task started even before the research on formal models for granularity (?, ?) and continued as a parallel stream of research (?, ?, ?, ?).

Finally, let us consider requirement c) above. Several inferencing operations have been defined on low-level representations, including equivalence, inclusion between granules in different granularities, and even complex inferencing services like constraint propagation (?). Even for simple operations no general method is available operating directly on the high level representation. Indeed, in some cases, the proposed methods cannot exploit the structure of the expression and require the enumeration of granules, which may be very inefficient. This is the case, for example, of the granule conversion methods presented by Ning e at. (?). Moreover, we are not aware of any method to perform other operations, such as equivalence or intersection of sets of granules, directly in terms of the high level representation.

The major goal of this paper is to provide a unique framework to satisfy the requirements a), b), and c) identified above, by adding to the existing results a smart and efficient technique to convert granularity specifications from the high-level algebraic formalism to the low-level one, for which many more reasoning tools are available. In particular, in this paper we focus on the conversion from the high-level formalism called *Calendar Algebra* (?) to the low-level formalism based on periodical sets (?, ?). Among the several proposals for the high-level (algebraic) specification of granularities, the choice of Calendar Algebra has two main motivations: first, it allows the user to express a large class of granularities; For a comparison of the expressiveness of Calendar Algebra with other formalisms see (?). Second, it provides the richest set of algebraic operations that are designed to reflect the intuitive ways in which users define new granularities. A discussion on the actual usability of this tool and on how it could be enhanced by a graphical user interface can be found in Section 6.2. The choice of the low-level formalism based on periodic-sets also has two main motivations: first, an efficient implementation of all the basic operations already exists and has been extensively experimented (?); second, it is the only one currently supporting the complex operations on granularities needed for constraint satisfaction, as it will be illustrated in more detail in Section 6.1.

The technical contribution of this paper is a hybrid algorithm that interleaves the conversion of calendar subexpressions into periodical sets with a step for period minimization. A central phase of our conversion procedure is to derive, for each algebraic subexpression, the periodicity of the output set. This periodicity is used to build the periodical representation of the subexpression that can be recursively used as operand of other expressions. Given a calendar algebra expression, the algorithm returns set-based granularity representations having minimal period length. The period length is the most relevant parameter for the performance both of basic operations on granularities and of more specialized ones like the operations used by the constraint satisfaction service. Extensive experimental work reported in this paper validates the techniques used in the algorithm, by showing, among other things, that (1) even large calendar expressions can be efficiently converted, and (2) less precise conversion formulas may lead to unacceptable computation time. This latter property shows the importance of carefully and accurately designed conversion formulas. Indeed, conversion formulas may seem trivial if the length of periodicity is not a concern. In designing our conversion formulas, we made an effort to reduce the period length of the resulting granularity representation, and thus render the whole conversion process computationally efficient.

In the next section we define granularities; several interesting relationships among them are highlighted and the periodical set representation is formalized. In Section 3 we define Calendar Algebra and present its operations. In Section 4 we describe the conversion process: after the definition of the three steps necessary for the conversion, for each algebraic operation we present the formulas to perform each step. In Section 5 we discuss the period minimality issue, and we report experimental results based on a full implementation of the conversion algorithm and of its extension ensuring minimality. In Section 6 we further motivate our work by presenting a complete application scenario. Section 7 reports the related work, and Section 8 concludes the paper.

2. Formal Notions of Time Granularities

Time granularities include very common ones like hours, days, weeks, months and years, as well as the evolution and specialization of these granularities for specific contexts or applications. Trading days, banking days, and academic semesters are just few examples of specialization of granularities that have become quite common when describing policies and constraints.

2.1 Time Granularities

A comprehensive formal study of time granularities and their relationships can be found in (?). In this paper, we only introduce notions that are essential to show our results. In particular, we report here the notion of *labeled granularity* which was proposed for the specification of a calendar algebra (?, ?); we will show later how any *labeled granularity* can be reduced to a more standard notion of granularity, like the one used by Bettini et al. (?).

Granularities are defined by grouping sets of instants into *granules*. For example, each granule of the granularity `day` specifies the set of instants included in a particular day. A label is used to refer to a particular granule. The whole set of time instants is called *time domain*, and for the purpose of this paper the domain can be an arbitrary infinite set with a total order relationship, $\leq$.

Definition 1 *A* labeled granularity *G is a pair $(\mathcal{L}_G, M)$, where $\mathcal{L}_G$ is a subset of the integers, and M is a mapping from $\mathcal{L}_G$ to the subsets of the time domain such that for each pair of integers i and j in $\mathcal{L}_G$ with $i < j$, if $M(i) \neq \emptyset$ and $M(j) \neq \emptyset$, then (1) each element in $M(i)$ is less than every element of $M(j)$, and (2) for each integer k in $\mathcal{L}_G$ with $i < k < j$, $M(k) \neq \emptyset$.*

The former condition guarantees the "monotonicity" of the granularity; the latter is used to introduce the bounds (see Section 2.2).

We call $\mathcal{L}_G$ the *label set* and for each $i \in \mathcal{L}_G$ we call $G(i)$ a *granule*; if $G(i) \neq \emptyset$ we call it a *non-empty granule*. When $\mathcal{L}_G$ is exactly the integers, the granularity is called "full-integer labeled". When $\mathcal{L}_G = \mathbb{Z}^+$ we have the same notion of granularity as used in several applications, e.g., (?). For example, following this labeling schema, if we assume to map `day`(1) to the subset of the time domain corresponding to January 1, 2001, `day`(32) would be mapped to February 1, 2001, `b-day`(6) to January 8, 2001 (the sixth business day), and `month`(15) to March 2002. The generalization to arbitrary label sets has been introduced mainly to facilitate conversion operations in the algebra, however our final goal is the conversion of a labeled granularity denoted by a calendar expression into a "positive-integer labeled" one denoted by a periodic formula.

2.2 Granularity Relationships

Some interesting relationships between granularities follows. The definitions are extended from the ones presented by Bettini et al. (?) to cover the notion of labeled granularity.

Definition 2 *If G and H are labeled granularities, then G is said to* group into *H, denoted $G \trianglelefteq H$, if for each non-empty granule $H(j)$, there exists a (possibly infinite) set S of labels of G such that $H(j) = \bigcup_{i \in S} G(i)$.*

Intuitively, $G \trianglelefteq H$ means that each granule of H is a union of some granules of G. For example, `day` $\trianglelefteq$ `week` since a week is composed of 7 days and `day` $\trianglelefteq$ `b-day` since each business day is a day.

Definition 3 *If G and H are labeled granularities, then G is said to be* finer than *H, denoted $G \preceq H$, if for each granule $G(i)$, there exists a granule $H(j)$ such that $G(i) \subseteq H(j)$.*

For example `business-day` is finer than `day`, and also finer than `week`.

We also say that G *partitions* H if $G \trianglelefteq H$ and $G \preceq H$. Intuitively G partitions H if $G \trianglelefteq H$ and there are no granules of G other than those included in granules of H. For example, both `day` and `b-day` group into `b-week` (business week, i.e., the business day in a week), but `day` does not partition `b-week`, while `b-day` does.

Definition 4 *A labeled granularity G_1 is a* label-aligned subgranularity *of a labeled granularity G_2 if the label set $\mathcal{L}_{G_1}$ of G_1 is a subset of the label set $\mathcal{L}_{G_2}$ of G_2 and for each i in $\mathcal{L}_{G_1}$ such that $G_1(i) \neq \emptyset$, we have $G_1(i) = G_2(i)$.*

Intuitively, G_1 has a subset of the granules of G_2 and those granules have the same label in the two granularities.

Granularities are said to be *bounded* when $\mathcal{L}_G$ has a first or last element or when $G(i) = \emptyset$ for some $i \in \mathcal{L}_G$. We assume the existence of an unbounded bottom granularity, denoted by $\perp$ which is full-integer labeled and groups into every other granularity in the system.

There are time domains such that, given any set of granularities, it is always possible to find a bottom one; for example, it can be easily proved that this property holds for each time domain that has the same cardinality as the integers. On the other hand, the same property does not hold for other time domains (e.g. the reals). However, the assumption about the existence of the bottom granularity is still reasonable since we address problems in which granularities are defined starting from a bottom one. The definition of a *calendar* as a set of granularities that have the same bottom granularity (?) captures this idea.

2.3 Granularity Conversions

When dealing with granularities, we often need to determine the granule (if any) of a granularity H that covers a given granule z of another granularity G. For example, we may wish to find the month (an interval of the absolute time) that includes a given week (another interval of the absolute time).

This transformation is obtained with the *up* operation. Formally, for each label $z \in \mathcal{L}_G$, $\lceil z \rceil_G^H$ is undefined if $\nexists z' \in \mathcal{L}_H$ s.t. $G(z) \subseteq H(z')$; otherwise, $\lceil z \rceil_G^H = z'$, where z' is the unique index value such that $G(z) \subseteq H(z')$. The uniqueness of z' is guaranteed by the monotonicity [1] of granularities. As an example, $\lceil z \rceil_{\texttt{second}}^{\texttt{month}}$ gives the month that includes the second z. Note that while $\lceil z \rceil_{\texttt{second}}^{\texttt{month}}$ is always defined, $\lceil z \rceil_{\texttt{week}}^{\texttt{month}}$ is undefined if week z falls between two months. Note that if $G \preceq H$, then the function $\lceil z \rceil_G^H$ is defined for each index value z. For example, since `day` $\preceq$ `week`, $\lceil z \rceil_{\texttt{day}}^{\texttt{week}}$ is always defined, i.e., for each day we can find the week that contains it. The notation $\lceil z \rceil^H$ is used when the source granularity can be left implicit (e.g., when we are dealing with a fixed set of granularities having a distinguished bottom granularity).

Another direction of the above transformation is the *down* operation: Let G and H be granularities such that $G \trianglelefteq H$, and z an integer. Define $\lfloor z \rfloor_G^H$ as the set S of labels of granules of G such that $\bigcup_{j \in S} G(j) = H(z)$.[2] This function is useful for finding, e.g., all the days in a month.

2.4 The Periodical Granules Representation

A central issue in temporal reasoning is the possibility of finitely representing infinite granularities. The definition of granularity provided above is general and expressive but it may be impossible to provide a finite representation of some of the granularities. Even labels (i.e., a subset of the integers) do not necessarily have a finite representation.

A solution has been first proposed by Bettini et al. (?). The idea is that most of the commonly used granularities present a periodical behavior; it means that there is a certain pattern that repeats periodically. This feature has been exploited to provide a method for finitely describing granularities. The formal definition is based on the *periodically groups into* relationship.

Definition 5 *A labeled granularity G* groups periodically into *a labeled granularity H ($G \trianglelefteq\!\!\!\!\!\sim H$) if $G \trianglelefteq H$ and there exist positive integers N and P such that*

(1) for each label i of H, $i+N$ is a label of H unless $i+N$ is greater than the greatest label of H, and

(2) for each label i of H, if $H(i) = \bigcup_{r=0}^{k} G(j_r)$ and $H(i+N)$ is a non-empty granule of H then $H(i+N) = \bigcup_{r=0}^{k} G(j_r + P)$, and

(3) if $H(s)$ is the first non-empty granule in H (if exists), then $H(s+N)$ is non-empty.

The *groups periodically into* relationship is a special case of the group into characterized by a periodic repetition of the "grouping pattern" of granules of G into granules of H. Its

1. Condition (1) of Definition 1.
2. This definition is different from the one given by Bettini et al (?) since it also considers non contiguous granules of G.

definition may appear complicated but it is actually quite simple. Since G groups into H, any granule $H(i)$ is the union of some granules of G; for instance assume it is the union of the granules $G(a_1), G(a_2), \ldots, G(a_k)$. Condition (1) ensures that the label $i+N$ exists (if it not greater than the greatest label of H) while condition (2) ensures that, if $H(i+N)$ is not empty, then it is the union of $G(a_1+P), G(a_2+P), \ldots, G(a_k+P)$. We assume that $\forall r = 0 \ldots k$, $(j_r + P) \in \mathcal{L}_G$; if not, the conditions are considered not satisfied. Condition (3) simply says that there is at least one of these repetitions.

We call each pair P and N in Definition 5, a *period length* and its associated *period label distance*. We also indicate with R the number of granules of H corresponding to each groups of P consecutive granules of $\perp$. More formally R is equal to the number of labels of H greater or equal than i and smaller than $i+N$ where i is an arbitrary label of H. Note that R is not affected by the value of i.

The period length and the period label distance are not unique; more precisely, we indicate with P_H^G the period length of H in terms of G and with N_H^G the period label distance of H in terms of G; the form P_H and N_H is used when $G = \perp$. Note that the period length is an integer value. For simplicity we also indicate with one period of a granularity H a set of R consecutive granules of H.

In general, the *periodically groups into* relationship guarantees that granularity H can be finitely described (in terms of granules of G).

Definition 6 *If $G \trianglelefteq H$, then H can be finitely described by providing: (i) a value for P and N; (ii) the set $\mathcal{L}^P$ of labels of H in one period of H; (iii) for each $a \in \mathcal{L}^P$, the finite set S_a of labels of G, such that $H(a) = \bigcup_{i \in S_a} G(i)$; (iv) the labels of first and last non-empty granules in H, if their values are not infinite.*

In this representation, the granules that have labels in $\mathcal{L}^P$ are the only ones that need to be explicitly represented; we call these granules the *explicit granules.*

If a granularity H can be represented as a periodic set of granules of a granularity G, then there exists an infinite number of pairs (P_H^G, N_H^G) for which the periodically groups into relation is satisfied. If the relation is satisfied for a pair (P, N), then it can be proved that it can also be satisfied for each pair $(\alpha P, \alpha N)$ with $\alpha \in \mathbb{N}^+$.

Definition 7 *A periodic representation of a granularity H in terms of G is called* minimal *if the period length P used in the representation has the smallest value among the period lengths appearing in all the pairs (P_H^G, N_H^G) for which H periodically groups into G.*

If H is fully characterized in terms of G, it is possible to derive the composition, in terms of G, of any granule of H. Indeed, if $\mathcal{L}^P$ is the set of labels of H with values in $\{b, \ldots, b + N_H^G - 1\}$, and we assume H to be unbounded, the description of an arbitrary granule $H(j)$ can be obtained by the following formula. Given $j' = [(j-1) \bmod N_H^G] + 1$ and

$$k = \begin{cases} \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor\right) \cdot N_H^G + j' & \text{if } \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor\right) \cdot N_H^G + j' \geq b \\[2ex] \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor + 1\right) \cdot N_H^G + j' & \text{otherwise} \end{cases}$$

we have

$$H(j) = \bigcup_{i \in S_k} G\left(P_H^G \cdot \left\lfloor \frac{j-1}{N_H^G} \right\rfloor + i - P_H^G \cdot \left\lfloor \frac{k-1}{N_H^G} \right\rfloor\right).$$

Example 1 *Figure 1 shows granularities* `day` *and* `week_parts` *i.e., the granularity that, for each week, contains a granule for the working days and a granule for the weekend. For the sake of simplicity, we denote* `day` *and* `week_parts` *with D and W respectively. Since $D \trianglelefteq W$, W is fully characterized in terms of D. Among different possible representations, in this example we decide to represent W in terms of D by $P_W^D = 7$, $N_W^D = 2$, $\mathcal{L}_W^P = \{3, 4\}$, $S_3 = \{8, 9, 10, 11, 12\}$ and $S_4 = \{13, 14\}$. The composition of each granule of W can then be easily computed; For example the composition of $W(6)$ is given by the formula presented above with $j' = 2$ and $k = 4$. Hence $W(6) = D(7 \cdot 2 + 13 - 7 \cdot 1) \cup D(7 \cdot 2 + 14 - 7 \cdot 1) = D(20) \cup D(21)$.*

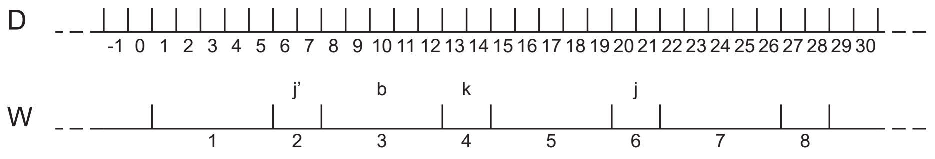


Figure 1: *Periodically groups into* example

3. Calendar Algebra

Several high-level symbolic formalisms have been proposed to represent granularities (?, ?).

In this work we consider the formalism proposed by Ning et al. (?) called *Calendar Algebra*. In this approach a set of algebraic operations is defined; each operation generates a new granularity by manipulating other granularities that have already been generated. The relationships between the operands and the resulting granularities are thus encoded in the operations. All granularities that are generated directly or indirectly from the bottom granularity form a *calendar*, and these granularities are related to each other through the operations that define them. In practice, the choices for the bottom granularity include `day`, `hour`, `second`, `microsecond` and other granularities, depending on the accuracy required in each application context.

In the following we illustrate the calendar algebra operations presented by Ning et al. (?) together with some restrictions introduced by Bettini et al. (?).

3.1 The Grouping-Oriented Operations

The calendar algebra consists of the following two kinds of operations: the *grouping-oriented operations* and the *granule-oriented operations*. The grouping-oriented operations group certain granules of a granularity together to form new granules in a new granularity.

3.1.1 The Grouping Operation

Let G be a full-integer labeled granularity, and m a positive integer. The grouping operation $Group_m(G)$ generates a new granularity G' by partitioning the granules of G into m-granule groups and making each group a granule of the resulting granularity. More precisely, $G' = \mathit{Group}_m(G)$ is the granularity such that for each integer i,

$$G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i\cdot m} G(j).$$

For example, given granularity `day`, granularity `week` can be generated by the calendar algebra expression `week` $= \mathit{Group}_7($`day`$)$ if we assume that `day`(1) corresponds to Monday, i.e., the first day of a week.

3.1.2 The Altering-tick Operation

Let G_1, G_2 be full-integer labeled granularities, and l, k, m integers, where G_2 partitions G_1, and $1 \leq l \leq m$. The altering-tick operation $\mathit{Alter}^m_{l,k}(G_2, G_1)$ generates a new granularity by periodically expanding or shrinking granules of G_1 in terms of granules of G_2. Since G_2 partitions G_1, each granule of G_1 consists of some contiguous granules of G_2. The granules of G_1 can be partitioned into m-granule groups such that $G_1(1)$ to $G_1(m)$ are in one group, $G_1(m+1)$ to $G_1(2m)$ are in the following group, and so on. The goal of the altering-tick operation is to modify the granules of G_1 so that the l-th granule of every m-granule group will have $|k|$ additional (or fewer when $k < 0$) granules of G_2. For example, if G_1 represents 30-day groups (i.e., $G_1 = \mathit{Group}_{30}($`day`$)$) and we want to add a day to every 3-rd month (i.e., to make March to have 31 days), we may perform $\mathit{Alter}^{12}_{3,1}($`day`$, G_1)$.

The altering-tick operation can be formally described as follows. For each integer i such that $G_1(i) \neq \emptyset$, let b_i and t_i be the integers such that $G_1(i) = \cup_{j=b_i}^{t_i} G_2(j)$ (the integers b_i and t_i exist because G_2 partitions G_1). Then $G' = \mathit{Alter}^m_{l,k}(G_2, G_1)$ is the granularity such that for each integer i, let $G'(i) = \emptyset$ if $G_1(i) = \emptyset$, and otherwise let

$$G'(i) = \bigcup_{j=b'_i}^{t'_i} G_2(j),$$

where

$$b'_i = \begin{cases} b_i + (h-1)\cdot k, & \text{if } i = (h-1)\cdot m + l, \\ b_i + h\cdot k, & \text{otherwise,} \end{cases}$$

$$t'_i = t_i + h\cdot k,$$

and

$$h = \left\lfloor \frac{i-l}{m} \right\rfloor + 1.$$

Example 2 *Figure 2 shows an example of the* `Alter` *operation. Granularity G_1 is defined by $G_1 = \mathit{Group}_5(G_2)$ and granularity G' is defined by $G' = \mathit{Alter}^2_{2,-1}(G_2, G_1)$, which means shrinking the second one of every two granules of G_1 by one granule of G_2.*

The original definition of altering-tick given by Ning et al. (?) as reported above, has the following problems when an arbitrary negative value for k is used: (1) It allows the definition of a G' that is not a full-integer labeled granularity and (2) It allows the

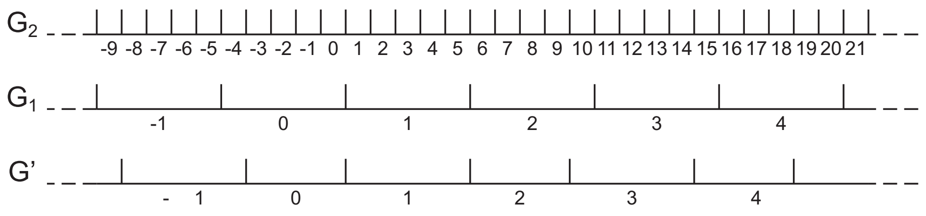


Figure 2: *Altering-tick* operation example

definition of a G' that does not even satisfy the definition of granularity. In order to avoid this undesired behavior, we impose the following restriction:

$$k > -(mindist(G1, 2, G2) - 1)$$

where $mindist()$ is formally defined by Bettini et al. (?).

Intuitively, $mindist(G1, 2, G2)$ represents the minimum distance (in terms of granules of $G2$) between two consecutive granules of $G1$.

3.1.3 The Shift Operation

Let G be a full-integer labeled granularity, and m an integer. The shifting operation $\mathit{Shift}_m(G)$ generates a new granularity G' by shifting the labels of G by m positions. More formally, $G' = \mathit{Shift}_m(G)$ is the granularity such that for each integer i, $G'(i) = G(i - m)$. Note that G' is also full-integer labeled.

3.1.4 The Combining Operation

Let G_1 and G_2 be granularities with label sets $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$ respectively. The combining operation $\mathit{Combine}(G_1, G_2)$ generates a new granularity G' by combining all the granules of G_2 that are included in one granule of G_1 into one granule of G'. More formally, for each $i \in \mathcal{L}_1$, let $s(i) = \emptyset$ if $G_1(i) = \emptyset$, and otherwise let $s(i) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$. Then $G' = \mathit{Combine}(G_1, G_2)$ is the granularity with the label set $\mathcal{L}_{G'} = \{i \in \mathcal{L}_{G_1} | s(i) \neq \emptyset\}$ such that for each i in $\mathcal{L}_{G'}$, $G'(i) = \bigcup_{j \in s(i)} G_2(j)$.

As an example, given granularities `b-day` and `month`, the granularity for business months can be generated by $\texttt{b-month} = \mathit{Combine}(\texttt{month}, \texttt{b-day})$.

3.1.5 The Anchored Grouping Operation

Let G_1 and G_2 be granularities with label sets $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$ respectively, where G_2 is a label-aligned subgranularity of G_1, and G_1 is a full-integer labeled granularity. The anchored grouping operation $\mathit{Anchored\text{-}group}(G_1, G_2)$ generates a new granularity G' by combining all the granules of G_1 that are between two granules of G_2 into one granule of G'. More formally, $G' = \mathit{Anchored\text{-}group}(G_1, G_2)$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_2}$ such that for each $i \in \mathcal{L}_{G'}$, $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ where i' is the next label of G_2 after i.

For example, each academic year at a certain university begins on the last Monday in August, and ends on the day before the beginning of the next academic year. Then, the granularity corresponding to the academic years can be generated by $AcademicYear = \mathit{Anchored\text{-}group}(\texttt{day}, \texttt{lastMondayOfAugust})$.

3.2 The Granule-Oriented Operations

Differently from the grouping-oriented operations, the granule-oriented operations do not modify the granules of a granularity, but rather enable the selection of the granules that should remain in the new granularity.

3.2.1 The Subset Operation

Let G be a granularity with label set $\mathcal{L}_G$, and m, n integers such that $m \leq n$. The subset operation $G' = \mathit{Subset}_m^n(G)$ generates a new granularity G' by taking all the granules of G whose labels are between m and n. More formally, $G' = \mathit{Subset}_m^n(G)$ is the granularity with the label set $\mathcal{L}_{G'} = \{i \in \mathcal{L}_G \mid m \leq i \leq n\}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G(i)$. For example, given granularity `year`, all the years in the 20th century can be generated by $\texttt{20CenturyYear} = \mathit{Subset}_{1900}^{1999}(\texttt{year})$. Note that G' is a label-aligned subgranularity of G, and G' is not a full-integer labeled granularity even if G is. We also allow the extensions of setting $m = -\infty$ or $n = \infty$ with semantics properly extended.

3.2.2 The Selecting Operations

The selecting operations are all binary operations. They generate new granularities by selecting granules from the first operand in terms of their relationship with the granules of the second operand. The result is always a label-aligned subgranularity of the first operand granularity.

There are three selecting operations: *select-down*, *select-up* and *select-by-intersect*. To facilitate the description of these operations, the $\Delta_k^l(S)$ notation is used. Intuitively, if S is a set of integers, $\Delta_k^l(S)$ selects l elements starting from the k-th one (for a formal description of the Δ operator see (?)).

Select-down operation. For each granule $G_2(i)$, there exits a set of granules of G_1 that is contained in $G_2(i)$. The operation $\mathit{Select\text{-}down}_k^l(G_1, G_2)$, where $k \neq 0$ and $l > 0$ are integers, selects granules of G_1 by using $\Delta_k^l(\cdot)$ on each set of granules (actually their labels) of G_1 that are contained in one granule of G_2. More formally, $G' = \mathit{Select\text{-}down}_k^l(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \cup_{i \in \mathcal{L}_{G_2}} \Delta_k^l(\{j \in \mathcal{L}_{G_1} \mid \emptyset \neq G_1(j) \subseteq G_2(i)\}),$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, Thanksgiving days are the fourth Thursdays of all Novembers; if `Thursday` and `November` are given, it can be generated by $\texttt{Thanksgiving} = \mathit{Select\text{-}down}_4^1(\texttt{Thursday}, \texttt{November})$.

Select-up operation. The select-up operation $\mathit{Select\text{-}up}(G_1, G_2)$ generates a new granularity G' by selecting the granules of G_1 that contain one or more granules of G_2. More formally, $G' = \mathit{Select\text{-}up}(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \{i \in \mathcal{L}_{G_1} | \exists j \in \mathcal{L}_{G_2} (\emptyset \neq G_2(j) \subseteq G_1(i)), \}$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, given granularities `Thanksgiving` and `week`, the weeks that contain Thanksgiving days can be defined by $\texttt{ThanxWeek} = \mathit{Select\text{-}up}(\texttt{week}, \texttt{Thanksgiving})$.

Select-by-intersect operation. For each granule $G_2(i)$, there may exist a set of granules of G_1, each intersecting $G_2(i)$. The *Select-by-intersect*$^l_k(G_1, G_2)$ operation, where $k \neq 0$ and $l > 0$ are integers, selects granules of G_1 by applying $\Delta^l_k(\cdot)$ operator to all such sets, generating a new granularity G'. More formally, $G' =$ *Select-by-intersect*$^l_k(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \cup_{i \in \mathcal{L}_{G_2}} \Delta^l_k(\{j \in \mathcal{L}_{G_1} \mid G_1(j) \cap G_2(i) \neq \emptyset\}),$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, given granularities `week` and `month`, the granularity consisting of the first week of each month (among all the weeks intersecting the month) can be generated by `FirstWeekOfMonth` $=$ *Select-by-intersect*$^1_1$(`week`, `month`).

3.2.3 The Set Operations

In order to have the set operations as a part of the calendar algebra and to make certain computations easier, we restrict the operand granularities participating in the set operations so that the result of the operation is always a valid granularity: the set operations can be defined on G_1 and G_2 only if there exists a granularity H such that G_1 and G_2 are both label-aligned subgranularities of H. In the following, we describe the union, intersection, and difference operations of G_1 and G_2, assuming that they satisfy the requirement.

Union. The union operation $G_1 \cup G_2$ generates a new granularity G' by collecting all the granules from both G_1 and G_2. More formally, $G' = G_1 \cup G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cup \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$,

$$G'(i) = \begin{cases} G_1(i), & i \in \mathcal{L}_1, \\ G_2(i), & i \in \mathcal{L}_2 - \mathcal{L}_1. \end{cases}$$

For example, given granularities `Sunday` and `Saturday`, the granularity of the weekend days can be generated by `WeekendDay` $=$ `Sunday` $\cup$ `Saturday`.

Intersection. The intersection operation $G_1 \cap G_2$ generates a new granularity G' by taking the common granules from both G_1 and G_2. More formally, $G' = G_1 \cap G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cap \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$ (or equivalently $G_2(i)$).

Difference. The difference operation $G_1 \setminus G_2$ generates a new granularity G' by excluding the granules of G_2 from those of G_1. More formally, $G' = G_1 \setminus G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \setminus \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$.

4. From Calendar Algebra to Periodical Set

In this section we first describe the overall conversion process and then we report the formulas specific for the conversion of each calendar algebra operation. Finally, we present a procedure for relabeling the resulting granularity, a sketch complexity analysis and some considerations about the period length minimality.

4.1 The Conversion Process

Our final goal is to provide a correct and effective way to convert calendar expressions into periodical representations. Under appropriate limitations, for each calendar algebra operation, if the periodical descriptions of the operand granularities are known, it is possible to compute the periodical characterization of the resulting granularity.

This result allows us to calculate, for any calendar, the periodical description of each granularity in terms of the bottom granularity. In fact, by definition, the bottom granularity is fully characterized; hence it is possible to compute the periodical representation of all the granularities that are obtained from operations applied to the bottom granularity. Recursively, the periodical description of all the granularities can be obtained.

The calendar algebra presented in the previous section can represent all the granularities that are periodical with finite exceptions (i.e., any granularity G such that bottom groups periodically with finite exceptions into G). Since with the periodical representations defined in Section 2 it is not possible to express the finite exceptions, we need to restrict the calendar algebra so that it cannot represent them. This implies allowing the *Subset* operation to be only used as the last step of deriving a granularity. Note that in the calendar algebra presented by Ning et al. (?) there was an extension to the altering-tick operation to allow the usage of ∞ as the m parameter (i.e., $G' = Alter_{l,k}^{\infty}(G_2, G_1)$); the resulting granularity has a single exception hence is not periodic. This extension is disallowed here in order to generate periodical granularities only (without finite exceptions).

The conversion process can be divided into three steps: in the first one the period length and period label distance are computed; in the second we derive the set $\mathcal{L}^P$ of labels in one period, and in the last one the composition of the explicit granules is computed. For each operation we identify the correct formulas and algorithms for the three steps.

The **first step** consists in computing the period length and the period label distance of the resulting granularity. Those values are calculated as a function of the parameters (e.g. the "grouping factor" m, in the *Group* operation) and the operand granularities (actually their period lengths and period label distances).

The **second step** in the conversion process is the identification of the label set of the resulting granularity. In Section 2.4 we pointed out that in order to fully characterize a granularity it is sufficient to identify the labels in any period of the granularity. In spite of this theoretical result, to perform the computations required by each operation we need the explicit granules of the operand granularities to be "aligned". There are two possible approaches: the first one consist in computing the explicit granules in any period and then recalculate the needed granules in the correct position in order to eventually align them. The second one consists in aligning all the periods containing the explicit granules with a fixed granule in the bottom granularity. After considering both possibilities, for performance reasons, we decided to adopt the second approach. We decided to use $\perp(1)$ as the "alignment point" for all the granularities. A formal definition of the used formalism follows.

Let G be a granularity and i be the smallest positive integer such that $\lceil i \rceil^G$ is defined. We call $l_G = \lceil i \rceil^G$ and $\overline{\mathcal{L}}_G$ the set of labels of G contained in $l_G \ldots l_G + N_G - 1$. Note that this definition of $\overline{\mathcal{L}}_G$ is an instance of the definition of $\mathcal{L}^P$ given in Section 2.4. The definition of $\overline{\mathcal{L}}_G$ provided here is useful for representing G and actually the final goal of this step is to compute $\overline{\mathcal{L}}_G$; however $\overline{\mathcal{L}}_G$ is not suitable for performing the computations. The problem is that if $G(l_G)$ starts before $\perp(1)$ (i.e., $min(\lfloor l_G \rfloor^G) < 1$) then the granule $G(l_G + N_G)$ begins at P_G or before P_G, and hence $G(l_G + N_G)$ is necessary for the computations; however $l_G + N_G \notin \overline{\mathcal{L}}_G$.

To solve the problem we introduce the symbol $\hat{\mathcal{L}}_G$ to represent the set of all labels of granules of G that cover one in $\perp(1) \dots \perp(P_G)$. It is easily seen that if $G(l_G)$ does not cover $\perp(0)$, then $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G$, otherwise $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G \cup \{l_G + N_G\}$. Therefore the conversion between $\overline{\mathcal{L}}$ and $\hat{\mathcal{L}}$ and vice versa is immediate.

The notion of $\hat{\mathcal{L}}$ is still not enough to perform the computations. The problem is that when a granularity G is used as an operand in an operation, the period length of the resulting granularity G' is generally bigger than the period length of G. Therefore it is necessary to extend the notion of $\hat{\mathcal{L}}_G$ to the period length $P_{G'}$ of G' using $P_{G'}$ in spite of P_G in the definition of $\hat{\mathcal{L}}$. The symbol used for this notion is $\hat{\mathcal{L}}_G^{P_{G'}}$.

The idea is that when G is used as the operand in an operation that generates G', $\hat{\mathcal{L}}_G^{P_{G'}}$ is computed from $\overline{\mathcal{L}}_G$. This set is then used by the formula that we provide below to compute $\overline{\mathcal{L}}_{G'}$.

The computation of $\overline{\mathcal{L}}_{G'}$ is performed as follows: if G' is defined by an operation that returns a full-integer labeled granularity, then it is sufficient to compute the value of l'_G. Indeed it is easily seen that $\overline{\mathcal{L}}_{G'} = \{i \in \mathbb{Z} | l'_G \leq i \leq l'_G + N_{G'} - 1\}$. If G' is defined by any other algebraic operation, we provide the formulas to compute $\hat{\mathcal{L}}_{G'}$; from $\hat{\mathcal{L}}_{G'}$ we easily derive $\overline{\mathcal{L}}_{G'}$.

Example 3 *Figure 3 shows granularities $\perp$, G and H; it is clear that $P_G = P_H = 4$ and $N_G = N_H = 3$. Moreover, $l_G = l_H = 6$ and therefore $\overline{\mathcal{L}}_G = \overline{\mathcal{L}}_H = \{6, 7\}$. Since $0 \notin \lfloor 6 \rfloor^G$ then $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G$. On the other hand, since $0 \in \lfloor 6 \rfloor^H$, then $\hat{\mathcal{L}}_H = \overline{\mathcal{L}}_H \cup \{6 + 3\}$.*

Suppose that a granularity G' has period length $P_{G'} = 8$; then $\hat{\mathcal{L}}_G^{P'_G} = \{6, 7, 9, 10\}$ and $\hat{\mathcal{L}}_H^{P_{G'}} = \{6, 7, 9, 10, 12\}$.

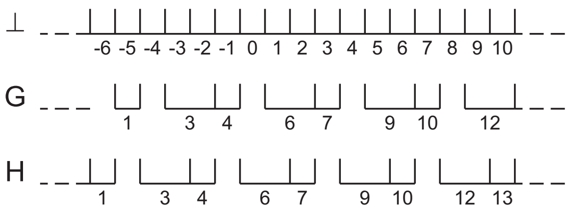


Figure 3: $\overline{\mathcal{L}}$, l, $\hat{\mathcal{L}}$ and $\hat{\mathcal{L}}^{P_{G'}}$ examples

The **third** (and last) **step** of the conversion process is the computation of the composition of the explicit granules. Once $\overline{\mathcal{L}}_{G'}$ has been computed, it is sufficient to apply, for each label of $\overline{\mathcal{L}}_{G'}$ the formulas presented in Chapter 3.

In Sections 4.3 to 4.10 we show, for each calendar algebra operation, how to compute the first and second conversion steps.

4.2 Computability Issues

In some of the formulas presented below it is necessary to compute the set S of labels of a granularity G such that $\forall i \in S\ G(i) \subseteq H(j)$ where H is a granularity and j is a specific label of H. Since $\mathcal{L}_G$ contains an infinite number of labels, it is not possible to check, $\forall i \in \mathcal{L}_G$ if $G(i) \subseteq H(j)$. However it is easily seen that $\forall i \in S\ \exists k$ s.t. $G(\lceil k \rceil^G) \subseteq H(j)$. Therefore $\forall i \in S\ \exists k$ s.t. $G(\lceil k \rceil^G)$ is defined and $k \in \lfloor j \rfloor^H$.

Therefore we compute the set S by considering all the labels i of $\mathcal{L}_G$ s.t. $\exists n \in \lfloor j \rfloor^H$ s.t. $\lceil n \rceil^G = i$ and $G(i) \subseteq H(j)$. Since the set $\lfloor j \rfloor^H$ is finite[3], the computation can be performed in a finite time. The consideration is analogous if S is the set such that $\forall i \in S\ G(i) \supseteq H(j)$ or $\forall i \in S\ (G(i) \cap H(j) \neq \emptyset)$.

4.3 The Group Operation

Proposition 1 *If $G' = Group_m(G)$, then:*

1. $P_{G'} = \frac{P_G \cdot m}{GCD(m, N_G)}$ *and* $N_{G'} = \frac{N_G}{GCD(m, N_G)}$*;*
2. $l_{G'} = \left(\left\lfloor \frac{l_G - 1}{m} \right\rfloor + 1 \right)$*;*
3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i \cdot m} G(j)$.

Example 4 *Figure 4 shows an example of the group operation: $G' = Group_3(G)$. Since $P_G = 1$ and $N_G = 1$, then $P_{G'} = 3$ and $N_G = 1$. Moreover, since $\overline{\mathcal{L}}_G = \{-7\}$, then $l_G = -7$ and therefore $l_{G'} = -2$ and $\overline{\mathcal{L}}_{G'} = \{-2\}$. Finally $G'(-2) = G(-8) \cup G(-7) \cup G(-6)$ i.e., $G'(-2) = \bot(0) \cup \bot(1) \cup \bot(2)$.*

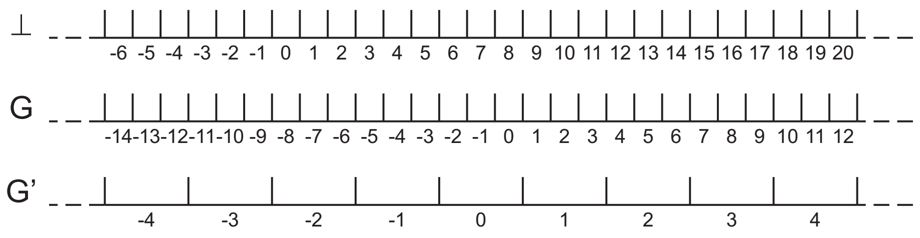


Figure 4: *Group* operation example

4.4 The Altering-tick Operation

Proposition 2 *If $G' = Alter^m_{l,k}(G_2, G_1)$ then:*

1.

$$N_{G'} = lcm\left(N_{G_1}, m, \frac{P_{G_2} \cdot N_{G_1}}{GCD(P_{G_2} \cdot N_{G_1}, P_{G_1})}, \frac{N_{G_2} \cdot m}{GCD(N_{G_2} \cdot m, |k|)}\right)$$

3. With the calendar algebra it is not possible to define granularities having granules that maps to an infinite set of time instants.

and

$$P_{G'} = \left(\frac{N_{G'} \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{N_{G'} \cdot k}{m} \right) \cdot \frac{P_{G_2}}{N_{G_2}}$$

2. $l_{G'} = \lceil l_{G_2} \rceil_{G_2}^{G'}$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j=b'_i}^{t'_i} G(j)$ *where* b'_i *and* t'_i *are defined in Section 3.1.2.*

Referring to step 2., note that when computing $l_{G'}$ the explicit characterization of the granules of G' is still unknown. To perform the operation $\lceil l_{G_2} \rceil_{G_2}^{G'}$ we need to know at least the explicit granules of one of its periods. We choose to compute the granules labeled by $1 \dots N_{G'}$. When $l_{G'}$ is derived, the granules labeled by $l_{G'} \dots l_{G'} + N_{G'} - 1$ will be computed so that the explicit granules are aligned to $\perp(1)$ as required.

Example 5 *Figure 5 shows an example of the altering-tick operation:* $G' = Alter^{3}_{2,1}(G_2, G_1)$. *Since* $P_{G_1} = 4$, $N_{G_1} = 1$, $P_{G_2} = 4$ *and* $N_{G_2} = 2$, *then* $N_{G'} = 6$ *and* $P_{G'} = 28$. *Moreover, since* $\overline{\mathcal{L}}_{G_2} = \{-10, -9\}$, *then* $l_{G_2} = -10$ *and therefore* $l_{G'} = \lceil -10 \rceil_{G_2}^{G'} = -4$ *and hence* $\overline{\mathcal{L}}_{G_2} = \{-4, -3, \dots, 0, 1\}$. *Finally* $G'(-4) = G_1(-11) \cup G_1(-10) \cup G_1(-9) = \perp(-1) \cup \perp(0) \cup \perp(1) \cup \perp(3) \cup \perp(4)$*; analogously we derive* $G'(-3)$, $G'(-2)$, $G'(-1)$, $G'(0)$ *and* $G'(1)$.

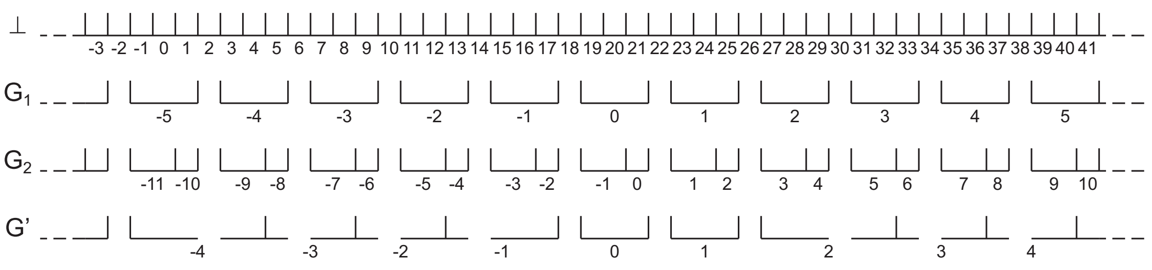


Figure 5: *Alter* operation example

4.5 The Shift Operation

Proposition 3 *If* $G' = Shift_m(G)$, *then:*

1. $P_{G'} = P_{G_1}$ *and* $N_{G'} = N_{G_1}$;

2. $l_{G'} = l_G + m$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = G(i - m)$.

Example 6 *The shifting operation can easily model time differences. Suppose granularity* `USEast-Hour` *stands for the hours of US Eastern Time. Since the hours of the US Pacific Time are 3 hours later than those of US Eastern Time, the hours of US Pacific Time can be generated by* `USPacific-Hour`$= Shift_{-3}($`USEast-Hour`$)$.

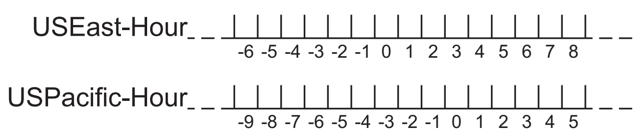


Figure 6: *Shift* operation example

4.6 The Combining Operation

Proposition 4 *Given $G' = Combining(G_1, G_2)$, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$;
2. $\forall i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ *let be* $\widetilde{s}(i) = \{j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$; *then* $\hat{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \widetilde{s}(i) \neq \emptyset\}$;
3. $\forall i \in \overline{\mathcal{L}}_{G'}$ $G'(i) = \bigcup_{j \in s(i)} G_2(j)$.

Example 7 *Figure 7 shows an example of the combining operation: $G' = Combine(G_1, G_2)$. Since $P_{G_1} = 6$, $N_{G_1} = 2$, $P_{G_2} = 4$ and $N_{G_2} = 2$, then $P_{G'} = 12$ and $N_{G'} = 4$. Moreover, since $\overline{\mathcal{L}}_{G_1} = \{1\}$ and $0 \in \lfloor 1 \rfloor^{G_1}$, then $\hat{\mathcal{L}}_{G_1} = \{1, 3\}$ and hence $\hat{\mathcal{L}}_{G_1}^{P_{G'}} = \{1, 3, 5\}$. Since $\tilde{s}(i) \neq \emptyset$ for $i \in \{1, 3, 5\}$, then $\hat{\mathcal{L}}_{G'} = \{1, 3, 5\}$; moreover, since $0 \in \lfloor 1 \rfloor^{G'}$, then $\overline{\mathcal{L}}_{G'} = \{1, 3\}$. Finally $s(1) = \{-1, 0\}$ and $s(3) = \{2, 3\}$; consequently, $G'(1) = G_2(-1) \cup G_2(0)$ i.e., $G'(1) = \bot(-1) \cup \bot(0) \cup \bot(1)$ and $G'(3) = G_2(2) \cup G_2(3)$ i.e., $G'(3) = \bot(4) \cup \bot(5) \cup \bot(7)$.*

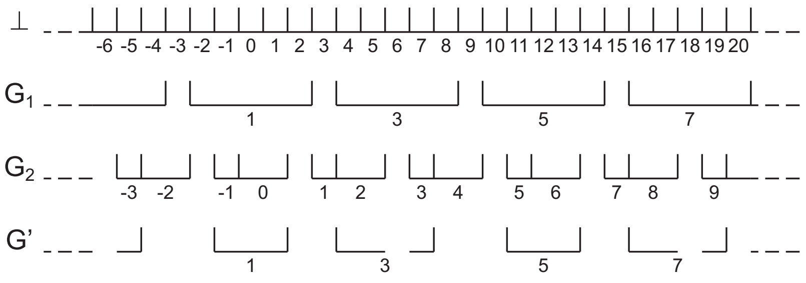


Figure 7: *Combine* operation example

4.7 The Anchored Grouping Operation

Proposition 5 *Given $G' = Anchored\text{-}group(G_1, G_2)$, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_2}}{P_{G_2}}$;
2.

$$\hat{\mathcal{L}}_{G'} = \begin{cases} \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{if } l_{G_2} = l_{G_1}, \\ \{l'_{G_2}\} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{otherwise,} \end{cases}$$

where l'_{G_2} is the greatest among the labels of $\mathcal{L}_{G_2}$ that are smaller than l_{G_2}.

3. $\forall i \in \overline{\mathcal{L}}_{G'}$ $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ *where* i' *is the next label of* G_2 *after* i.

Example 8 *Figure 8 shows an example of the anchored grouping operation: the* `USweek` *(i.e., a week starting with a* `Sunday`*) is defined by the operation* Anchored-group(`day`, `Sunday`)*. Since* $P_{\texttt{day}} = 1$ *and* $P_{\texttt{Sunday}} = 7$*, then the period length of* `USweek` *is 7. Moreover since* $l_{\texttt{day}} = 11$*,* $l_{\texttt{Sunday}} = 14$ *and* $\hat{\mathcal{L}}_{\texttt{Sunday}}^{P_{\texttt{USweek}}} = \{14\}$*, then* $\hat{\mathcal{L}}_{\texttt{USweek}} = \{7\} \cup \{14\}$*. Clearly, since* $0 \in \lfloor 7 \rfloor^{\texttt{USweek}}$ *then* $\overline{\mathcal{L}}_{\texttt{USweek}} = \{7\}$*. Finally,* $\texttt{USweek}(7) = \bigcup_{j=7}^{13} \texttt{day}(j) = \bigcup_{k=-3}^{3} \perp(k)$*.*

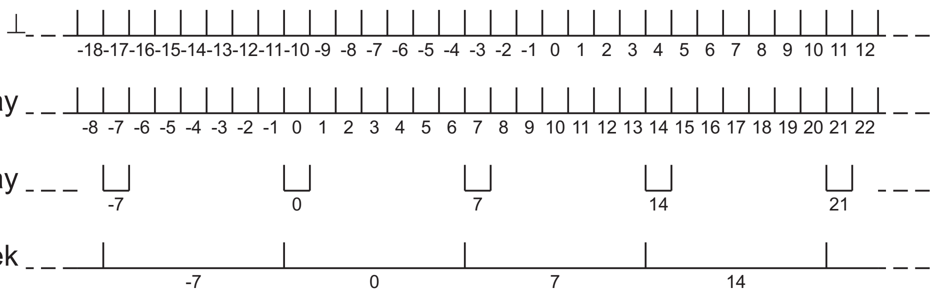


Figure 8: *Anchored Grouping* operation example

4.8 The Subset Operation

The *Subset* operation only modifies the operand granularity by introducing the bounds. The period length, the period label distance, $\overline{\mathcal{L}}$ and the composition of the explicit granules are not affected.

4.9 The Selecting Operations

4.9.1 The Select-down Operation

Proposition 6 *Given* $G' = \textit{Select-down}_k^l(G_1, G_2)$*, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_1}}{P_{G_1}}$;

2. $\forall i \in \mathcal{L}_{G_2}$ *let*
$$A(i) = \Delta_k^l \left(\{j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i)\}\right).$$
Then
$$\hat{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}} \left\{a \in A(i) | a \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}\right\};$$

3. $\forall i \in \overline{\mathcal{L}}_{G'}$ $G'(i) = G_1(i)$.

Example 9 *Figure 9 shows an example of the Select-down operation in which granularity* G' *is defined as:* $G' = \textit{Select-down}_2^1(G_1, G_2)$*. Since* $P_{G_1} = 4$*,* $N_{G_1} = 2$ *and* $P_{G_2} = 6$

then $P_{G'} = 12$ *and* $N_{G'} = 6$. *Moreover, since* $\overline{\mathcal{L}}_{G_2} = \{-3\}$ *and* $0 \in \lfloor -3 \rfloor^{G_2}$, *then* $\hat{\mathcal{L}}_{G_2} = \{-3, -2\}$ *and* $\hat{\mathcal{L}}_{G_2}^{P_{G'}} = \{-3, -2, -1\}$. *Intuitively,* $A(-3) = \{-5\}$, $A(-2) = \{-2\}$ *and* $A(-1) = \{1\}$. *Hence* $\hat{\mathcal{L}}_{G'} = \{-5, -2, 1\}$ *and therefore, since* $0 \in \lfloor -5 \rfloor^{G'}$, $\overline{\mathcal{L}}_{G'} = \{-5, -2\}$. *Finally* $G'(-5) = G_1(-5) = \perp(0) \cup \perp(1)$ *and* $G'(-2) = G_1(-2) = \perp(6)$.

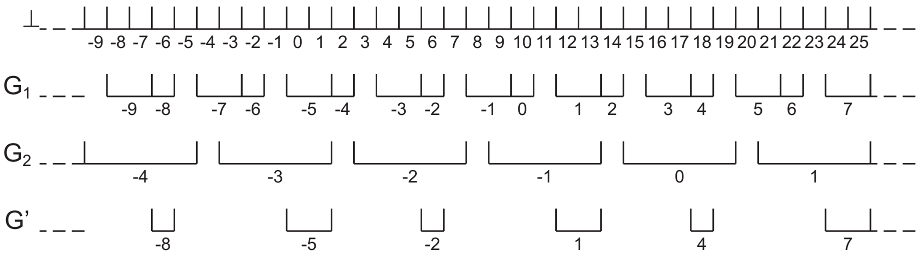


Figure 9: *Select-down* operation example

4.9.2 The Select-up Operation

Proposition 7 *Given* $G' = \textit{Select-up}(G_1, G_2)$, *then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_1}}{P_{G_1}}$;
2. $$\hat{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \exists j \in \mathcal{L}_{G_2} \; s.t. \, \emptyset \neq G_2(j) \subseteq G_1(i)\};$$
3. $\forall i \in \overline{\mathcal{L}}_{G'} \; G'(i) = G_1(i)$.

Example 10 *Figure 10 shows an example of the Select-up operation:* $G' = \textit{Select-up}(G_1, G_2)$. *Since* $P_{G_1} = 6$, $N_{G_1} = 3$ *and* $P_{G_2} = 4$ *then* $P_{G'} = 12$ *and* $N_{G'} = 6$. *Moreover, since* $\overline{\mathcal{L}}_{G_1} = \{-3, -2, -1\}$ *and* $0 \in \lfloor -3 \rfloor^{G_2}$, *then* $\hat{\mathcal{L}}_{G_1} = \{-3, -2, -1, 0\}$ *and* $\hat{\mathcal{L}}_{G_1}^{P'_G} = \{-3, -2, -1, 0, 1, 2, 3\}$. *Since* $G_1(-3) \supseteq G_2(-6)$, $G_1(-1) \supseteq G_2(-4)$ *and* $G_1(3) \supseteq G_2(0)$ *then* $\hat{\mathcal{L}}_{G'} = \{-3, -1, 3\}$ *and, since* $0 \in \lfloor -3 \rfloor^{G'}$, *then* $\overline{\mathcal{L}}_{G'} = \{-3, 1\}$ *Finally* $G'(-3) = G_1(-3) = \perp(0) \cup \perp(1)$ *and* $G'(-1) = G_1(-1) = \perp(4)$.

4.9.3 The Select-by-intersect Operation

Proposition 8 *Given* $G' = \textit{Select-by-intersect}_k^l(G_1, G_2)$, *then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$;
2. *then* $\forall i \in \mathcal{L}_{G_2}$ *let*
$$A(i) = \Delta_k^l \left(\{j \in \mathcal{L}_{G_1} | G_1(j) \cap G_2(i) \neq \emptyset\}\right).$$
then
$$\hat{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}} \left\{a \in A(i) | a \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}\right\}.$$

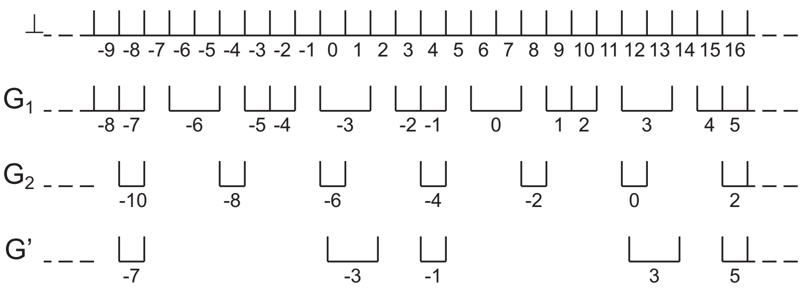


Figure 10: *Select-up* operation example

3. $\forall i \in \overline{\mathcal{L}}_{G'}$ $G'(i) = G_1(i)$.

Example 11 *Figure 11 shows an example of the Select-by-intersect operation in which* $G' =$ *Select-by-intersect*$_2^1(G_1, G_2)$. *Since* $P_{G_1} = 4$, $N_{G_1} = 2$ *and* $P_{G_2} = 6$ *then* $P_{G'} = 12$ *and* $N_{G'} = 6$. *Moreover, since* $\overline{\mathcal{L}}_{G_2} = \{-3\}$ *and* $0 \in \lfloor -3 \rfloor^{G_2}$, *then* $\hat{\mathcal{L}}_{G_2} = \{-3, -2\}$ *and* $\hat{\mathcal{L}}_{G_2}^{P_{G'}} = \{-3, -2, -1\}$. *Intuitively,* $A(-3) = \{-6\}$, $A(-2) = \{-2\}$ *and* $A(-1) = \{0\}$. *Hence* $\hat{\mathcal{L}}_{G'} = \{-2, 0\}$ *and therefore, since* $0 \notin \lfloor -5 \rfloor^{G'}$, *then* $\overline{\mathcal{L}}_{G'} = \{-2, 0\}$. *Finally* $G'(-2) = G_1(-2) = \perp(6)$ *and* $G'(0) = G_1(0) = \perp(10)$.

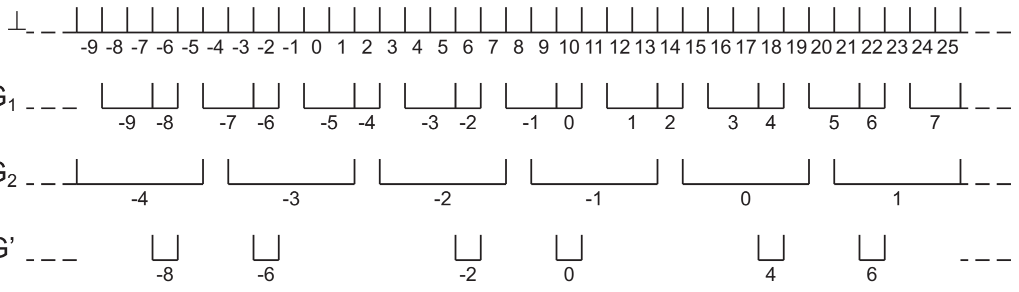


Figure 11: *Select-by-intersect* operation example

4.10 The Set Operations

Since a set operation is valid if the granularities used as argument are both labeled aligned granularity of another granularity, the following property is used.

Proposition 9 *If* G *is a labeled aligned subgranularity of* H, *then* $\frac{N_G}{P_G} = \frac{N_H}{P_H}$.

Proposition 10 *Given* $G' = G_1 \cup G_2$, $G'' = G_1 \cap G_2$ *and* $G''' = G_1 \setminus G_2$, *then:*

1. $P_{G'} = P_{G''} = P_{G'''} = lcm(P_{G_1}, P_{G_2})$ *and*
   $N_{G'} = N_{G''} = N_{G'''} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$;

2. $\hat{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G_1}^{P_{G'}} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}$; $\hat{\mathcal{L}}_{G''} = \hat{\mathcal{L}}_{G_1}^{P_{G''}} \cap \hat{\mathcal{L}}_{G_2}^{P_{G''}}$; $\hat{\mathcal{L}}_{G'''} = \hat{\mathcal{L}}_{G_1}^{P_{G'''}} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'''}}$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \begin{cases} G_1(i), & i \in \mathcal{L}_{G_1} \\ G_2(i), & \textit{otherwise}, \end{cases}$

$\forall i \in \overline{\mathcal{L}}_{G''}\ G''(i) = G_1(i)$ *and* $\forall i \in \overline{\mathcal{L}}_{G'''}\ G'''(i) = G_1(i)$

Example 12 *Figure 12 shows an example of the set operations. Note that both* G_1 *and* G_2 *are labeled aligned subgranularities of* H*. Then* $G' = G_1 \cup G_2$*,* $G'' = G_1 \cap G_2$ *and* $G''' = G_1 \setminus G_2$*. Since* $P_{G_1} = P_{G_2} = 6$ *and* $N_{G_1} = N_{G_2} = 6$ *then* $P_{G'} = P_{G''} = P_{G'''} = 6$ *and* $N_{G'} = N_{G''} = N_{G'''} = 2$*. Moreover, since* $\hat{\mathcal{L}}_{G_1} = \{1, 2\}$ *and* $\hat{\mathcal{L}}_{G_2} = \{2, 3\}$*, then* $\hat{\mathcal{L}}_{G'} = \{1, 2, 3\}$*,* $\hat{\mathcal{L}}_{G''} = \{2\}$ *and* $\hat{\mathcal{L}}_{G'''} = \{1\}$*. Finally* $G'(1) = G_1(1)$*,* $G'(2) = G_1(2)$ *and* $G'(3) = G_2(3)$*;* $G''(2) = G_1(2)$ *and* $G'''(1) = G_1(1)$*.*

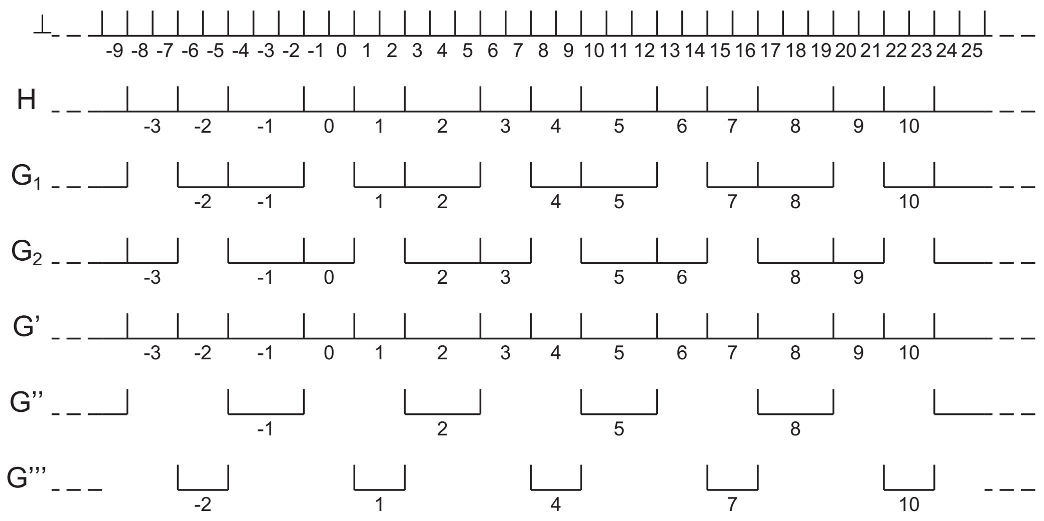


Figure 12: *Set* operations example

4.11 Relabeling

Granularity processing algorithms are much simpler if restricted to operate on full-integer labeled granularities. Moreover, a further simplification is obtained by using only the positive integers as the set of labels (i.e., $\mathcal{L} = \mathbb{Z}^+$).

In this section we show how to relabel a granularity G to obtain a full-integer labeled granularity G'. A granularity G'' such that $\mathcal{L}_{G''} = \mathbb{Z}^+$ can be obtained by using $G'' = Subset_1^{\infty}(G')$

Note that with the relabeling process some information is lost: for example, if G is a labeled aligned subgranularity of H and $G \neq H$, then, after the relabeling, G is not a labeled aligned subgranularity of H. The lost information is semantically meaningful in the calendar algebra, and therefore the relabeling must be performed only when the granularity will not be used as an operator in an algebraic operation.

Let G be a labeled granularity, i and j integers with $i \in \mathcal{L}_G$ s.t. $G(i) \neq \emptyset$. The relabeling operation $Relabel_i^j(G)$ generates a full-integer labeled granularity G' by relabeling $G(i)$ as $G'(j)$ and relabel the next (and previous) granule of G by the next (and previous, respectively) integer. More formally, for each integer k, if $k = j$, then let $G'(k) = G(i)$, and otherwise let $G'(k) = G(i')$ where $G(i')$ is the $|j-k|$-th granule of G after (before, respectively) $G(i)$. If the required $|j-k|$-th granule of G does not exist, then let $G'(k) = \emptyset$. Note the G' is always a full-integer labeled granularity.

The relabeling procedure can be implemented in the periodic representation we adopted by computing the value of $l_{G'}$. It is easily seen that once $l_{G'}$ is known, the full characterization of G' can be obtained with: $P_{G'} = P_G$; $N_{G'} = R_{G'} = R_G$ and $\overline{\mathcal{L}}_{G'} = \{l_{G'}, l_{G'}+1, \ldots, l_{G'}+N_{G'}-2, l_{G'}+N_{G'}-1\}$. It is clear that the explicit representation of the granules is not modified.

To compute $l_{G'}$ consider the label $i' = i - \left\lfloor \frac{i-l_G}{N_G} \right\rfloor \cdot N_G$; i' represents the label of $\overline{\mathcal{L}}_G$ such that $i-i'$ is a multiple of N_G. Therefore it is clear that the label $j' \in \overline{\mathcal{L}}_{G'}$ s.t. $G'(j') = G(i')$ can be computed by $j' = j - \left\lfloor \frac{i-l_G}{N_G} \right\rfloor \cdot N_{G'}$. Finally $l_{G'}$ is obtained with $l_{G'} = j' - |\delta|$ where δ is the distance, in terms of number of granules of G, from $G(l_G)$ to $G(i')$.

Example 13 *Figure 13 shows an example of the Relabel operation: $G' = Relabel^4_{33}(G)$. Since $P_G = 4$ and $R_G = 2$ then $P_{G'} = 4$ and $N_{G'} = 2$. Moreover, $i' = 33 - \lfloor \frac{33-6}{5} \rfloor \cdot 5 = 8$ and $j' = 4 - \lfloor \frac{33-6}{5} \rfloor \cdot 2 = -6$. Since $l_G = 6$ and $i' = 8$ then $G(i')$ is the next granule of G after $G(l_G)$. Then $\delta = 1$ and hence $l_{G'} = -6 - 1 = -7$. It follows that $\overline{\mathcal{L}}_{G'} = \{-7, -6\}$. Finally $G'(-7) = G(6)$ and $G'(-6) = G(8)$.*

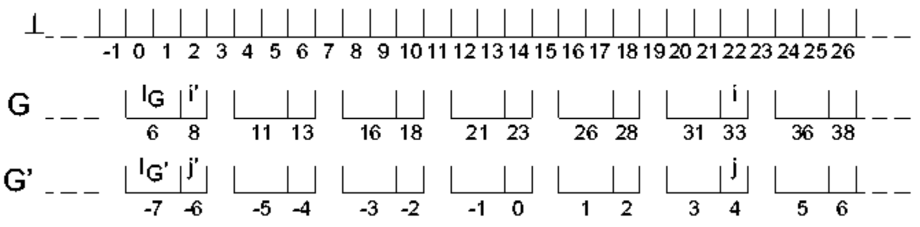


Figure 13: *Relabeling* example

The GSTP constraint solver imposes that the first non-empty granule of any granularity ($\perp$ included) is labeled with 1. Therefore, when using the relabeling operation for producing granularities for GSTP, the parameter j must be set to 1. The parameter i has to be equal to the smallest label among those that identify granules of G covering granules of $\perp$ that are all labeled with positive values. By definition of l_G, $i = l_G$ if $min(\lfloor l_G \rfloor^G) > 0$; otherwise i is the next label of G after l_G.

4.12 Complexity Issues

For each operation the time necessary to perform the three conversion steps, depends on the operation parameters (e.g. the "grouping factor" m, in the *Group* operation) and on the operand granularities (in particular the period length, the period label distance and the number of granules in one period).

A central issue is that if an operand granularity is not the bottom granularity, then its period is a function of the periods of the granularities that are the operands in the operation that defines it. For most of the algebraic operations, in the worst case the period of the resulting granularity is the product of the periods of the operands granularity.

For all operations, the **first step** in the conversion process can be performed in a constant or logarithmic time. Indeed the formulas necessary to derive the period length and the period label distance involve (i) standard arithmetic operations, (ii) the computation of the Greatest Common Divisor and (iii) the computation of the least common multiple. Part (i) can be computed in a constant time while (ii) and (iii) can be computed in a logarithmic time using Euclid's algorithm.

For some operations, the **second step** can be performed in constant time (e.g. *Group*, *Shift* or *Anchored-group*) or in linear time (e.g. set operations). For the other operations it is necessary to compute the set S of labels of a granularity G such that $\forall i \in S\ G(i) \subseteq H(j)$ where H is a granularity and $j \in \mathcal{L}_H$ (analogously if S is the set such that $\forall i \in S\ G(i) \supseteq H(j)$ or $\forall i \in S\ (G(i) \cap H(j) \neq \emptyset)$). This computation needs to be performed once for each granule $i \in P_H^{P_{G'}}$. The idea of the algorithm for solving the problem has been presented in Section 4.2. Several optimizations can be applied to that algorithm, but in the worst case (when H covers the entire time domain) it is necessary to perform a number of $\lceil \cdot \rceil^G$ operations linear in the period length of the resulting granularity. If an optimized data structure is used to represent the granularities, the $\lceil \cdot \rceil^G$ operation can be performed in constant time [4], then the time necessary to perform the second step is linear in the period length of the resulting granularity ($O(P_{G'})$).

The **last step** in the conversion process is performed in linear time with respect to the number of granules in a period of G'.

The complexity analysis of the conversion of a general algebraic expression needs to consider the composition of the operations and hence their complexity. Finally, relabeling, can be done in linear time.

A more detailed complexity analysis is out of the scope of this work.

5. Minimal Representation and Experimental Results

In this section we address the problem of guaranteeing that the converted representation is minimal in terms of the period length. As we will show in Example 14 the conversion formulas proposed in this paper do not guarantee a minimal representation of the result and it is not clear if conversion formulas ensuring minimality exist. Our approach is to apply a minimization step in the conversion.

The practical applicability of the minimization step depends on the period length of the representation that is to be minimized. Indeed, in our tests we noted that the minimization step is efficient if the conversion formulas proposed in Section 4 are adopted, while it is impractical when the conversion procedure returns a period that is orders of magnitude higher than the minimal one as would be the case if conversion formulas were constructed in a naive way.

4. If a non-optimized data structure is used, $\lceil \cdot \rceil^G$ requires logarithmic time.

5.1 Period Length Minimization

As stated in Section 2, each granularity can have different periodical representations and, for a given granularity, it is possible to identify a set of representations that are *minimal* i.e. adopting the smallest period length.

Unfortunately, the conversions do not always return a minimal representation, as shown by Example 14.

Example 14 *Consider a calendar that has* `day` *as the bottom granularity. We can define* `week` *as* $\texttt{week} = Group_7(\texttt{day})$*; by applying the formulas for the Group operation we obtain* $P_{\texttt{week}} = 7$ *and* $N_{\texttt{week}} = 1$*.*

We can now apply the Altering-tick operation to add one day to every first week every two weeks. Let this granularity be $G_1 = Alter^2_{1,1}(\texttt{day}, \texttt{week})$*; applying the formulas for the Altering-tick operation we obtain* $P_{G_1} = 15$ *and* $N_{G_1} = 2$*.*

We can again apply the Altering-tick operation to create a granularity G_2 *by removing one day from every first granule of* G_1 *every two granules of* G_1*:* $G_2 = Alter^2_{1,-1}(\texttt{day}, G_1)$*. Intuitively, by applying this operation we should get back to the granularity* `week`*, however using the formulas for the Altering-tick operation we obtain* $P_{G_2} = 14$ *and* $N_{G_2} = 2$*; Hence* G_2 *is not minimal.*

In order to qualitatively evaluate how close to the minimal representations the results of our conversions are, we performed a set of tests using an algorithm (?) for minimality checking. In our experimental results the conversions of algebraic expressions defining granularities in real-world calendars, including many user-defined non-standard ones, always returned exactly minimal representations. Non-minimal ones could only be obtained by artificial examples like the one presented in Example 14.

Although a non-minimal result is unlikely in practical calendars, the minimality of the granularity representation is known to greatly affect the performance of the algorithms for granularity processing, e.g., granularity constraint processing (?), calendar calculations (?), workflow temporal support (?). Hence, we considered an extension of the conversion algorithm by adding a minimization step exploiting the technique illustrated by Bettini et al. (?) to derive a minimal representation.

The choice of using only the conversion algorithm or the extended one with minimizations, should probably be driven by performance considerations. In Section 5.3 we report the results of our experiments showing that generally it is advantageous to apply the minimization step. In our implementation, presented in Section 5.2, it is possible to specify if the minimization step should be performed.

5.2 Implementation of the *CalendarConverter* Web Service

The conversion formulas presented in Section 4 have been implemented into the *Calendar-Converter* web service that converts Calendar Algebra representations into the equivalent periodical ones. More precisely, given a calendar in which granularities are expressed by Calendar Algebra operations, the service converts each operation into an equivalent periodical representation.

The service first rewrites each calendar algebra expression in order to express it only in terms of the bottom granularity. For example, if the bottom granularity is `hour`, the expression $\texttt{Monday} = \textit{Select-down}_1^1(\texttt{day}, \texttt{week})$ is changed to

$$\texttt{Monday} = \textit{Select-down}_1^1(\textit{Group}_{24}(\texttt{hour}), \textit{Group}_7(\textit{Group}_{24}(\texttt{hour})))$$

Then, Procedure 1 is run for each granularity's expression. The idea is that the periodical representation of each subexpression is recursively computed starting from the expressions having the bottom granularity as operand. Once each operand of a given operation has been converted to periodical representation, the corresponding formula presented in Section 4 is applied. We call this step the *ConvertOperation* procedure.

A trivial optimization of Procedure 1 consists in caching the results of the conversions of each subexpression so that it is computed only once, even if the subexpression appears several times (like $\textit{Group}_{24}(\texttt{hour})$ in the above `Monday` definition).

Procedure 1 ConvertExpression

- **Input**: a calendar algebra expression *ex*; a boolean value *minimize* that is set to **true** if the minimization step is to be executed;
- **Output**: the periodical representation of *ex*;
- **Method**:

```
if (ex is the bottom granularity) then
    return the periodical representation of the bottom granularity
end if
operands := $\emptyset$
for (each operand op of ex) do
    add ConvertExpression(op, minimize) to operands;
end for
result :=ConvertOperation(ex.getOperator(), operands)
if (minimize) then
    minimize the periodical representation of result
end if
return result;
```

5.3 Experimental Results

Our experiments address two main issues: first, we evaluate how the conversion formulas impact on the practical applicability of the conversion procedure and, second, we evaluate how useful is the minimization step.

For the first issue, we execute the conversion procedure with two different sets of conversion formulas and compare the results. The first set is laid out in Section 4. The other, that is less optimized, is taken from the preliminary version of this paper (?).

Table 1 shows that when converting calendars having granularities with small minimal period length (first two rows), using the formulas in Section 4 improves the performance by one order of magnitude; However, conversions and minimizations are almost instantaneous

Table 1: Impact of the conversion formulas on the performance of the conversion and minimization procedures (time in milliseconds).

Calendar		Section 4 formulas			Less optimized formulas		
Period	Bot	Conv.	Min.	Tot.	Conv.	Min.	Tot.
1 year	day	4	2	6	62	32	94
4 years	day	7	2	9	76	55	131
1 year	hour	9	2	11	2,244	126,904	129,148
4 years	hour	16	4	20	4,362	908,504	912,866
100 years	day	127	9	136	3,764	1,434,524	1,438,288

with both approaches. On the contrary, when the minimal period length is higher, (last three rows) the time required to minimize the periodical representation is up to five orders of magnitude larger if the formulas proposed by Bettini et al. (?) are used; as a consequence, the entire conversion may require several minutes while, using the formulas presented in Section 4, it still requires only a fraction of a second. If the period length is even larger, the conversion procedure is impractical if the formulas presented by Bettini et al. (?) are used, and indeed in our experiments we did not obtain a result in less than thirteen hours.

For the second issue, we perform a set of three experiments. In the first one we compare the performance of the conversion procedure with the performance of the minimization step. In the experiment we consider the case in which the conversion procedure produces minimal representations. In this case the minimization step is always an overhead since it cannot improve the performance of the conversion procedure.

Figure 14 shows the result of the experiment. Four calendars are considered, each one containing a set of granularities of the Gregorian calendar. The four calendars differs in the values of two parameters: the bottom granularity (it is `second` for cal-1 and cal-3 while it is `minute` for cal-2 and cal-4) and the period in which leap years and leap years exceptions are represented (it is 1, 4, 100 and 400 years for cal-1, cal-3, cal-2 and cal-4 respectively); As a consequence, the minimal period length of the granularities `month` and `year` is about $3 \cdot 10^7$ for cal-1, $5 \cdot 10^7$ for cal-2, 10^8 for cal-3 and $2 \cdot 10^8$ for cal-4.

As can be observed in Figure 14, the ratio between the time required to perform the conversions and the time required for the minimization step varies significantly from a minimum of 3% for cal-4 to a maximum of 23% for cal-3. The reason is that the complexity of the conversion procedure is mainly affected by the period length of the granularity having the largest period length. On the other hand, the complexity of the minimization step is affected also by other features of the granularities such as their internal structure and the number of integers that can divide at the same time the period label distance, the period length and the number of granules in one period; For more details see (?).

In the second experiment we consider the case in which the conversion procedure produces a non-minimal representation for a granularity in the input calendar; in this case it is possible to benefit from the minimization step. For example, suppose that a granularity G

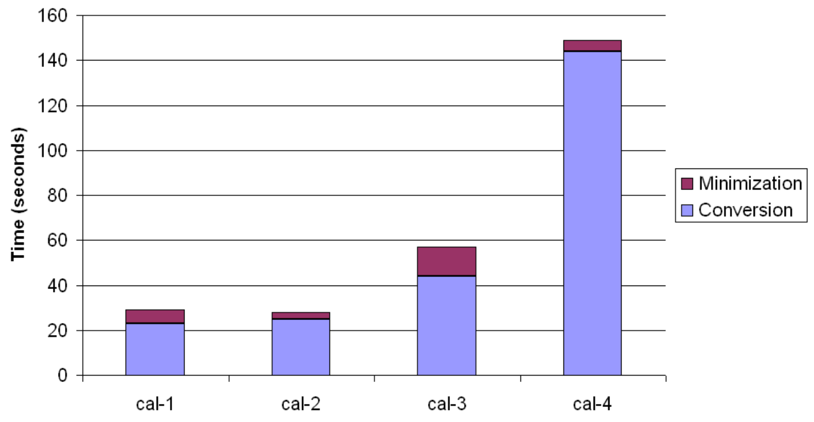


Figure 14: Impact of minimization over conversion; minimal conversions case.

is converted and that it is then used as an argument of another Calendar Algebra operation that defines a granularity H. The time required to compute the periodical representation of H strongly depends on the period length of G; If the period length of G is reduced by the execution of the minimization step, the conversion of H can be executed faster.

We produced this situation using a technique similar to the one of Example 14; we created Calendar Algebra definitions of the Gregorian calendar in which the granularity `day` is converted into a granularity having a non-minimal representation. Figure 15 shows the performance obtained converting the same granularities that were used in Figure 14. The difference was that in this case the definition of the granularity `day` is such that, after the conversion procedure, its period is twice as large as the minimal one (i.e., 48 hours or 2880 minutes or 172800 seconds depending on the bottom granularity that is used). It can be easily seen that in this case the use of the minimization step can improve the performance of the entire algorithm. Indeed, when the minimization step is performed, the conversion procedure requires about one half of the time that is required when no minimization is performed.

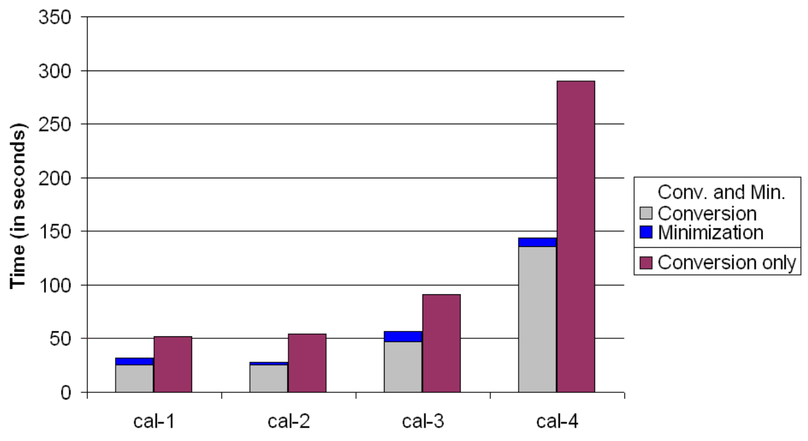


Figure 15: Impact of minimization over conversion; non-minimal case.

In the third experiment we evaluate the impact of the minimal representation on the performance of applications involving intensive manipulations of granularities. In the test we use the GSTP solver as such an application; it computes solutions of temporal constraints with granularities. A description of the architecture of the GSTP system is provided in Section 6.1.

Figure 16 shows our experiments performed on four temporal constraint networks with granularities. The four networks differs in the number of variables, in the number of constraints and in the granularities used to express the constraints. The networks labeled as "non-minimal" use granularities definitions that are obtained with a technique similar to the one used in Example 14, and have a period that is twice as large as the minimal one.

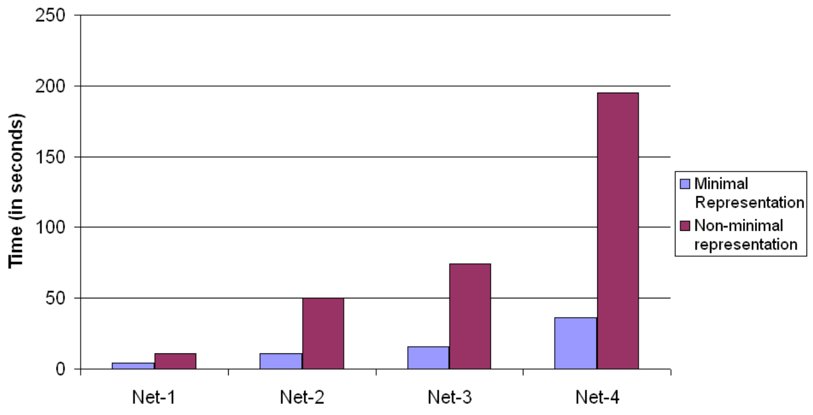


Figure 16: Impact of minimal representations on the performance of the GSTP solver.

Figure 16 shows that the use of minimal representations greatly improves the performance of the GSTP solver. Indeed in our experiments the ratio between the time required to solve the network using a non-minimal representation and a minimal one is between three and five. Moreover, the more time required to solve the network, the greater the improvement obtained using the minimal representation; this means that for very complex temporal networks we expect the improvement to be even higher.

Considering the results of our experiments, we conclude that, in general, it is advisable to perform the minimization step. In particular, it is very advantageous in the specific case of GSTP, based on the following considerations: i) the time required to perform the minimization step is only a fraction of the time required to perform the conversion procedure, ii) the conversions are performed off-line in most cases, with respect to granularity processing, and conversion results are cached for future use, and iii) the period length strongly influences the GSTP processing time that is in most cases much longer than the time needed for conversion.

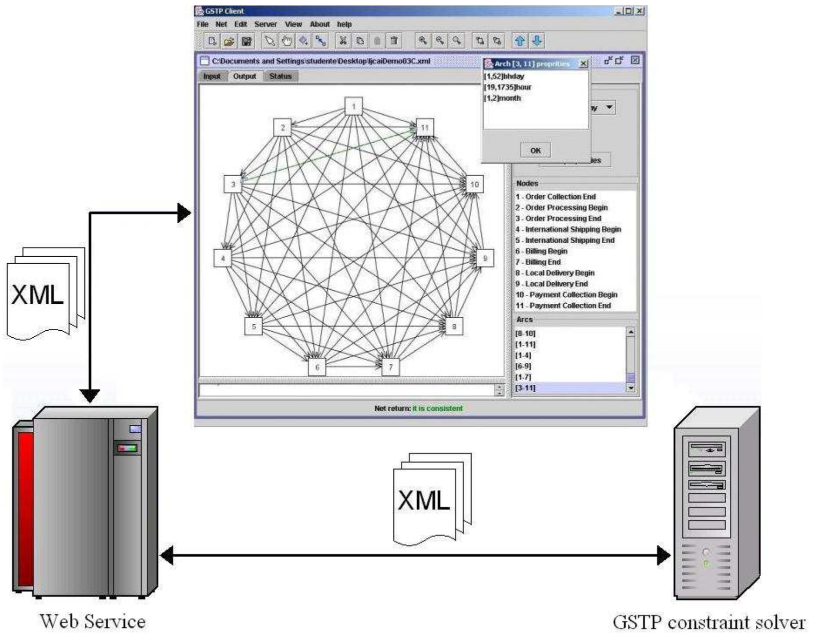
6. Applications

In this section we complement the motivations for this work with a sketch of the applications enabled by the proposed conversion. Firstly we describe the GSTP system, as an example of applications involving intensive manipulation of time granularities. GSTP is used to check the consistency and to find solutions of temporal constraint satisfaction problems with granularities[5]; It has also been applied to check the consistency of inter-organizational workflow models (?). Then, we discuss the use of Calendar Algebra to define new granularities that may later be part of the input of reasoning services, such as GSTP.

6.1 The GSTP System

The GSTP system has been developed at the University of Milan with the objective of providing universal access to the implementation of a set of algorithms for multi-granularity temporal constraint satisfaction (?). It allows the user to specify binary constraints of the form $Y - X \in [m, n]G$ where m and n are the minimum and maximum values of the distance from Y to X in terms of granularity G. Variables take values in the positive integers, and unary constraints can be applied on their domains. For example, the constraint: *Event2 should occur 2 to 4 business days after the occurrence of Event1* can be modeled by $Occ_{E2} - Occ_{E1} \in [2, 4]BDay$. This problem is considered an extension of STP (?) to multiple and arbitrary granularities. To our knowledge, GSTP is the only available system to solve this class of temporal constraint satisfaction problems.

Figure 17 shows the general architecture of the GSTP system. There are three main modules: the constraint solver; the web service, which enables external access to the solver; and a user interface that can be used locally or remotely to design and analyze constraint networks.



[1,52]bhday
[19,1735]hour
[1,2]month

Nodes
1 - Order Collection End
2 - Order Processing Begin
3 - Order Processing End
4 - International Shipping Begin
5 - International Shipping End
6 - Billing Begin
7 - Billing End
8 - Local Delivery Begin
9 - Local Delivery End
10 - Payment Collection Begin
11 - Payment Collection End

Arcs
[8-10]
[1-11]
[1-4]
[6-9]
[1-7]
[3-11]

Net return: it is consistent

Figure 17: The GSTP Architecture

The constraint solver is the C implementation of the ACG algorithm which has been proposed by Bettini et al. (?), and it runs on a server machine. Following the approach of

5. For a detailed description of the system, see (?).

Bettini et al. (?), the solver uses the representation of granularities based on periodical sets. This representation makes it possible to efficiently compute the core operations on granularities that are required to solve the constraint satisfaction problem. These operations involve, for example, the union and the intersection of periodical sets. While we cannot exclude that these operations may be computed in terms of alternative low level representations, it seems much harder to obtain similar results if a high level representation, such as Calendar Algebra, is used.

The second module of the system is the Web Service that defines, through a WSDL specification, the parameters that can be passed to the constraint solver, including the XML schema for the constraint network specification.

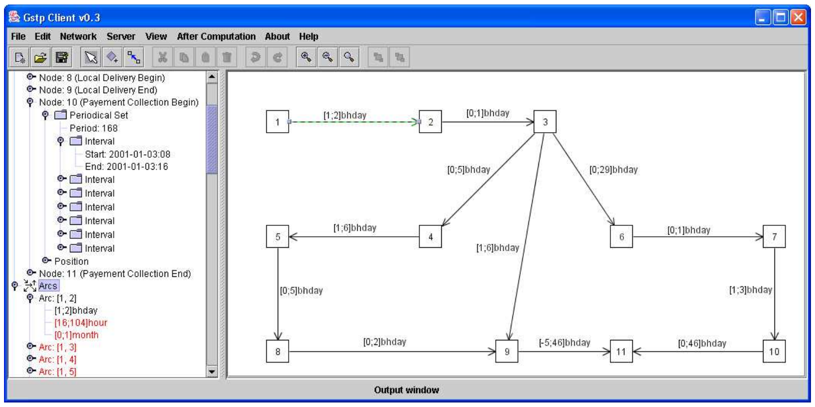


Figure 18: The GSTP User Interface

The third module is a remote Java-based user interface, which allows the user to easily edit constraint networks, to submit them to the constraint solver, and to analyze results. In particular, it is possible to have views in terms of specific granularities, to visualize implicit constraints, to browse descriptions of domains, and to obtain a network solution. Fig. 18 shows a screenshot from the interface.

6.2 Defining New Granularities

While the GSTP solver can handle arbitrary granularities, new granularities must be added by editing their explicit periodical representation. This is true in general for any multi-granularity reasoning service based on a low-level representation of granularities, and it is a painful task when the granularities have a large period. For example, in the experimental results illustrated in Figure 16, we used a representation of the granularity `month` that considers leap years and leap years exceptions in a period of 400 years. In this case, the users have to specify the representation of 4800 granules i.e., the number of months in 400 years.

Because the period length of real world granularities is generally high, a graphical interface does not help if it only supports the user to individually select the explicit granules. An effective solution requires the use of implicit or explicit operations on granules. Among the various proposals, Calendar Algebra provides the richest set of such operators. A question arises: is the definition of granularities in terms of Calendar Algebra really simpler than the specification of the periodical representation? Calendar Algebra does not seem to be user friendly: the exact semantics of each operator may not be immediate for an inexperienced user and some time is required in order to learn how to use each operator.

In practice, we do not think that it is reasonable to ask an unexperienced user to define granularities by writing Calendar Algebra expressions. Nevertheless, we do think that Calendar Algebra can be used by specialized user interfaces to guide the user when specifying granularities. In this sense, we believe that Calendar Algebra plays the same role that SQL does in the definition of databases queries. Similarly to Calendar Algebra, SQL is an abstraction tool that can be directly exploited in all its expressive power by an advanced user, but can also be used by a less experienced user through a graphical user interface, possibly with a reduced expressiveness.

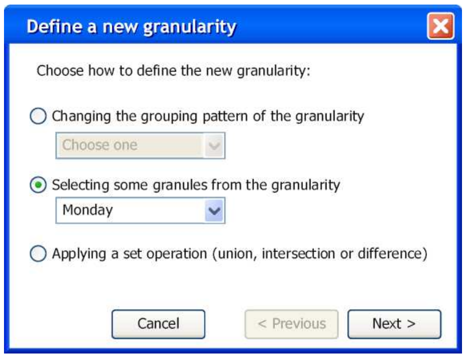
As mentioned above, in the case of periodical representations, graphical user interfaces are not sufficient for making the specification of new granularities practical. On the contrary, in the case of Calendar Algebra, user interfaces can strongly enhance the usability of Calendar Algebra, making its practical use possible also for the definition of involved granularities. There are at least two reasons for this difference. Firstly, the main difficulty of Calendar Algebra is the understanding of the semantics of the operators and the choice of the most appropriate one for a given task. An effective user interface can hide the existence of the algebraic operators to the user showing only how the operators modify existing granularities (i.e., the semantics of the operators). Secondarily, Calendar Algebra allows the compact definition of granularities. This is due to the fact that the Calendar Algebra operations are specifically designed to reflect the intuitive ways in which users define new granularities.

Example 15 shows how a graphical user interface can be effectively used to define a new granularity in terms of Calendar Algebra expression.

Example 15 *This example shows how a graphical user interface can be used to support the user in the definition of the granularity `final` as the set of days, each one corresponding to the last Monday of every academic semester. We assume that the granularities `Monday` and `academicSemester` have already been defined. The graphical user interface that we use in this example is a wizard that guides the user step by step. In the first step (Figure 19(a)) the user chooses the kind of operation he wants to perform. In the second step (Figure 19(b)) the user can provide more details about how he wants to modify the operand granularity (`Monday`, in the example). The results of this choice is a Calendar Algebra expression that is shown in the third step (Figure 19(c)); in this last window the user can also give a name to the granularity that has been defined.*

6.3 The Global Architecture

Figure 20 shows a possible architecture for the integration of GSTP, the interface for new granularity definitions and the CalendarConverter web service. A granularity repository collects the Calendar Algebra definitions. Upon request by the GSTP system definitions are converted in low-level representation by the CalendarConverter web service to be efficiently processed. Clearly, caching techniques can be used to optimize the process.


Define a new granularity

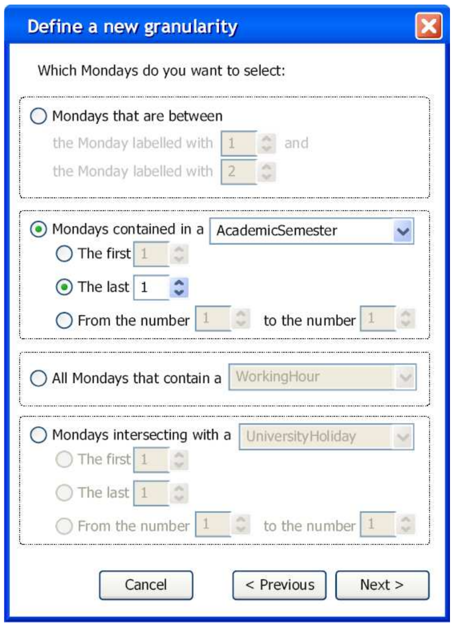
Choose how to define the new granularity:

○ Changing the grouping pattern of the granularity: Choose one

[✓ Selecting some granules from the granularity]: Monday

○ Applying a set operation (union, intersection or difference)

(a) Step 1.


Define a new granularity

Which Mondays do you want to select:

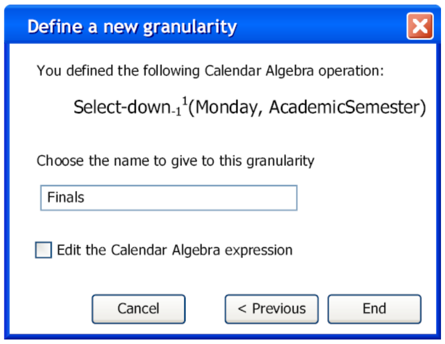
○ Mondays that are between the Monday labelled with 1 and the Monday labelled with 2

[✓ Mondays contained in a] AcademicSemester
○ The first 1
[✓ The last] 1
○ From the number 1 to the number 1

○ All Mondays that contain a WorkingHour

○ Mondays intersecting with a UniversityHoliday
○ The first 1
○ The last 1
○ From the number 1 to the number 1

(b) Step 2.


Define a new granularity

You defined the following Calendar Algebra operation:

$\text{Select-down}_{-1}^{1}(\text{Monday}, \text{AcademicSemester})$

Choose the name to give to this granularity: Finals

☐ Edit the Calendar Algebra expression

(c) Step 3.

Figure 19: A 3-steps wizard for visually defining a granularity using Calendar Algebra

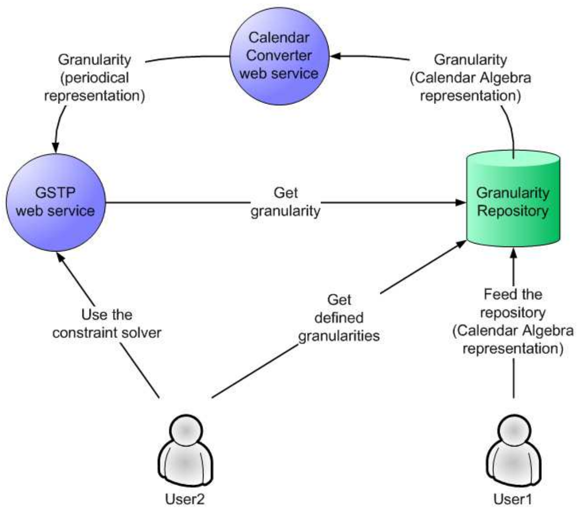

Calendar Converter web service
Granularity (periodical representation)
Granularity (Calendar Algebra representation)
GSTP web service
Get granularity
Granularity Repository
Use the constraint solver
Get defined granularities
Feed the repository (Calendar Algebra representation)
User2
User1

Figure 20: Integration of GSTP and CalendarConverter web services

7. Related Work

Several formalisms have been proposed for symbolic representation of granularities and periodicity. Periodicity and its application in the AI and DB area have been extensively investigated (?, ?, ?, ?). Regarding symbolic representation, it is well known the formalism proposed by Leban et al. (?), that is based on the notion of *collection*, and it is intended to represent temporal expressions occurring in natural language. A *collection* is a structured set of time intervals where the order of the collection gives a measure of the structure depth: an order 1 collection is an ordered list of intervals, and an order n ($n > 1$) collection is an ordered list of collections having order $n-1$. Two operators, called *slicing* and *dicing* are used to operate on collections by selecting specific intervals or sub-collections, and by further dividing an interval into a collection, respectively. For example, `Weeks:during:January2006` divides the interval corresponding to `January2006` into the intervals corresponding to the weeks that are fully contained in that month. This formalism has been adopted with some extensions by many researchers in the AI (?, ?) and Database area (?, ?). In particular, the control statements `if-then-else` and `while` have been introduced by Chandra et al. (?) to facilitate the representation of certain sets of intervals. For example, it is possible to specify: *the fourth Saturday of April if not an holiday, and the previous business day otherwise.*

As for the deductive database community, a second influential proposal is the *slice* formalism introduced by Niezette et al. (?). A slice denotes a (finite or infinite) set of not necessarily consecutive time intervals. For example, the slice `all.Years + {2,4}.Months + {1}.Days` ▷ `2.Days` denotes a set of intervals corresponding to the first 2 days of February and April of each year.

A totally different approach is the *calendar algebra* described by Ning et al. (?), and considered in this paper. The representation is based on a rich set of algebraic operators on periodic sets as opposed to *slicing* and *dicing* over nonconvex intervals.

None of the above cited papers provide a mapping to identify how each operator changes the mathematical characterization of the periodicity of the argument expressions. The problem of finding these mappings is not trivial for some operators.

In (?) the expressive power of the algebras proposed by Leban et al. (?) and Niezette et al. (?) is compared and an extension to the first is proposed in order to capture a larger set of granularities. Since the periodical representation is used to compare expressiveness, a mapping from calendar expressions in those formalisms to periodical representations can be found in the proofs of that paper. However, since minimality is not an issue for the purpose of comparing expressiveness, in many cases the mapping returns non-minimal representations.

Regarding alternative approaches for low-level representation, we already mentioned that the ones based on strings (?) and automata (?, ?) may be considered as an alternative for the target of our conversion. As a matter of fact, an example of the conversion of a Calendar Algebra expression into a string based representation can be found in (?). A complete conversion procedure appeared during the revision process of this paper in the PhD Dissertation by Puppis (?). The aim of the conversion is to prove that the *granspecs* formalism, used to represent granularities in terms of automata, has at least the same expressiveness as the Calendar Algebra. Hence, obtaining minimal representations was not the goal. Moreover, in their case minimization is not in terms of the period length, but in terms of the automaton size and automaton complexity. About the complexity of reasoning, given an automaton M, the worst case time complexity of the operations analogous to our *up* and *down* depends linearly on $||M||$, a value computed from M itself and called *complexity of* M. In this sense $||M||$ has the same role of our period length (P), even if a precise relationship between the two values is hard to obtain. In our approach we compute *up* in logarithmic time with respect to P and *down* in linear time with respect to the dimension of the result (that is bounded by P). Other operations, like checking for equivalence, seem to be more complex using automata (?). Techniques for minimization in terms of automaton complexity are presented by Dal Lago et al. (?), and the time complexity is proved to be polynomial, even if the exact bound is not explicitly given. In our approach, the worst case time complexity for the minimization is $O(P^{\frac{3}{2}})$ (?). Overall, the automata approach is very elegant and well-founded, but, on one side it still misses an implementation in order to have some experimental data to compare with, and on the other side only basic operations have been currently defined; it would be interesting to investigate the definition on that formalism of more complex operations like the ones required by GSTP.

8. Conclusion and Future Work

We have presented an hybrid algorithm that interleaves the conversion of Calendar Algebra subexpressions into periodical sets with the minimization of the period length. We have proved that the algorithm returns set-based granularity representations having minimal period length, which is extremely important for the efficiency of operations on granularities. Based on the technical contribution of this paper, a software system is being developed allowing users to access multi-granularity reasoning services by defining arbitrary time granularities with a high-level formalism. Our current efforts are mainly devoted to completing and refining the development of the different modules of the architecture shown in Section 6.3.

As a future work, we intend to develop effective graphical user interfaces to support the definition of Calendar Algebra expressions in a user friendly way. Example 15 described one of the possible interfaces. Another open issue is how to convert a periodical representation of a granularity into a "user friendly" Calendar Algebra expression. This conversion could be useful, for example, to present the result of a computation performed using the periodical representation. However, a naive conversion may not be effective since the resulting calendar algebra expression could be as involved as the periodical representation from which it is derived. For example, a conversion procedure is presented by Bettini et al. (?) to prove that the Calendar Algebra is at least as expressive as the periodical representation; however, the resulting Calendar Algebra expression is composed by a number of Calendar Algebra operations that is linear in the number of granules that are in one period of the original granularity. On the contrary, an effective conversion should generate Calendar Algebra expressions that are compact and easily readable by the user. This problem is somehow related to the discovery of calendar-based association rules (?). Finally, we intend to investigate the usage of the automaton-based representation as a low-level granularity formalism. It would be interesting to know whether, using this representation, it is possible to compute the same operations that can be computed with the periodical representation and if any performance gain could be achieved.

Acknowledgments

We thank the anonymous referees for their useful comments and suggestions. The work of Bettini and Mascetti was partially supported by Italian MIUR InterLink project N.II04C0EC1D. The work of Wang was partially supported by the US NSF grant 5-24376.

Appendix A. Proofs

A.1 Transitivity of the *Periodically Groups Into* Relationship

In order to prove the correctness of the conversions of algebraic expressions into periodical sets, it is useful to have a formal result about the transitivity of the *periodically groups into* relation. In addition to transitivity of $\trianglelefteq$, Theorem 1 also says something about period length values.

Theorem 1 *Let G and H be two unbounded granularities such that G is periodic in terms of the bottom granularity (i.e., $\perp \trianglelefteq G$) and H is periodic in terms of G (i.e., $G \trianglelefteq H$). Let P_H^G and N_H^G be the period length and the period label distance of H in terms of granules of G, and N_G the period label distance of G in terms of $\perp$. Then, if $P_H^G = \alpha N_G$ for some positive integer α, then H is periodic in terms of the bottom granularity (i.e., $\perp \trianglelefteq H$) and $P_H = \alpha P_G$.*

Proof. Since by hypothesis $G \trianglelefteq H$ and $P_H^G = \alpha N_G$, $\forall i$ if $H(i) = \bigcup_{r=0}^{n_i} G(i_r)$, then $H(i + N_H^G) = \bigcup_{r=0}^{n_i} G(i_r + \alpha N_G)$. This can be also written as follows:
if

$$H(i) = G(i_0) \cup ... \cup G(i_{n_i}) \tag{1}$$

then $\exists \beta \in \mathbb{N} s.t.$:

$$H(i + N_H^G) = G(i_0 + \alpha N_G) \cup ... \cup G(i_{n_i} + \alpha N_G) \tag{2}$$

Since $\perp \trianglelefteq G$, if

$$G(i_j) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k}) \tag{3}$$

then

$$G(i_j + N_G) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k} + P_G) \tag{4}$$

This can be clearly extended using αN_G instead of N_G.

$$G(i_j + \alpha N_G) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k} + \alpha P_G) \tag{5}$$

Rewriting (1) substituting $G(i_j)$ according to (3) and rewriting (2) substituting $G(i_j + \alpha N_G)$ according to (5), we obtain:
if $H(i) = \underbrace{\perp(i_{0,0}) \cup \ldots \cup \perp(i_{0,\tau_{i_0}})}_{G(i_0)} \cup \ldots \cup \underbrace{\perp(i_{n_i,0}) \cup ... \cup \perp(i_{n_i,\tau_{i_{n_i}}})}_{G(i_{n_i})}$

then $H(i+N_H^G) = \underbrace{\perp(i_{0,0}+\alpha P_G)\cup\ldots\cup\perp(i_{0,\tau_{i_0}}+\alpha P_G)}_{G(i_0+\alpha N_G)}\cup\ldots$
$\cup\underbrace{\perp(i_{n_i,0}+\alpha P_G)\cup\ldots\cup\perp(i_{n_i,\tau_{in_i}}+\alpha P_G)}_{G(i_{n_i}+\alpha N_G)}$
Hence the second condition of Definition 5 is satisfied. The third one is always satisfied for unbounded granularities. The first one is satisfied too; in fact since $G \trianglelefteq H$ with a period label distance of N_H^G, then for each label i of H, $i+N_H^G$ is a label of H. Hence, by definition of periodically-groups-into $\perp \trianglelefteq H$ with $P_H = \alpha P_G$ and $N_H = N_H^G$. □

A.2 Proof of Proposition 1

A.2.1 Part 1

From the definition of the *Group* operation, for all $i \in \mathbb{N}$:

$$G'(i) = \bigcup_{j=(i-1)m+1}^{im} G(j) = G(im-m+1)\cup\ldots\cup G(im) = G(\lambda)\cup\ldots\cup G(\lambda+m-1)$$

with $\lambda = im-m+1$. Furthermore, $\forall k \in \mathbb{N}$:

$$G'(i+k) = \bigcup_{j=(i+k-1)m+1}^{(i+k)m} G(j) = G(im+km-m+1)\cup\ldots\cup G(im+km) =$$

$$= G(\lambda+km)\cup\ldots\cup G(\lambda+km+m-1)$$

Hence,

$$\text{If } G'(i') = \bigcup_{r=0}^{m-1} G(\lambda+r) \text{ then } G'(i'+k) = \bigcup_{r=0}^{m-1} G(\lambda+r+km). \tag{6}$$

This holds for each k. If we use $k = \frac{N_G}{GCM(m,N_G)}$ (note that $k \in \mathbb{N}$), then all the hypotheses of Theorem 1 are satisfied: (i) $\perp \trianglelefteq G$ (by hypothesis); (ii) $G \trianglelefteq G'$ (since $G \trianglelefteq G'$, $\mathcal{L}_{G'} = \mathbb{Z}$, and (6) holds); (iii) $P_{G'}^G = \frac{m\cdot N_G}{GCM(m,N_G)}$ (since we use $k = \frac{N_G}{GCM(m,N_G)}$ and, from (6) we know that $P_{G'}^G = km$). Therefore, by Theorem 1, $\perp \trianglelefteq G'$ with $P_{G'} = \frac{mP_G}{GCM(m,N_G)}$ and $N_{G'} = \frac{N_G}{GCM(m\cdot N_G)}$.

A.2.2 Part 2

By definition of l, we need to show that $G'\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right) = \bigcup_{j=b}^{t} G(j)$ with $b \le l_G \le t$.

From the definition of the *Group* operation, $G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i\cdot m} G(i)$; hence:

$$G'\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right) = \bigcup_{j=\left\lfloor\frac{l_G-1}{m}\right\rfloor\cdot m+1}^{\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right)\cdot m} G(j)$$

We prove the thesis showing that (1) $\left\lfloor \frac{l_G-1}{m} \right\rfloor \cdot m+1 \le l_G$ and that (2) $\left(\left\lfloor \frac{l_G-1}{m} \right\rfloor + 1\right) \cdot m \ge l_G$.

(1) Since $\left\lfloor \frac{l_G-1}{m} \right\rfloor \le \frac{l_G-1}{m}$, hence $\left\lfloor \frac{l_G-1}{m} \right\rfloor \cdot m + 1 \le l_G$

(2) First we prove that $\left\lfloor \frac{l_G-1}{m} \right\rfloor \ge \frac{l_G}{m} - 1$. Since $\left\lfloor \frac{l_G-1}{m} \right\rfloor = \frac{l_G-1-[(l_G-1) mod\ m]}{m}$ we have to prove that $\frac{l_G-1-[(l_G-1) mod\ m]}{m} \ge \frac{l_G}{m} - 1$; it is equivalent to the inequality $-[(l_G-1) mod\ m] \ge -m+1$ that is true since $(l_G-1) mod\ m \le m-1$. Since $\left\lfloor \frac{l_G-1}{m} \right\rfloor \ge \frac{l_G}{m} - 1$ it is trivial that $\left(\left\lfloor \frac{l_G-1}{m} \right\rfloor + 1\right) \cdot m \ge l_G$.

A.3 Proof of Proposition 2

A.3.1 Part 1

Proof sketch

We show that $G_2 \trianglelefteq G'$ with $P_{G'}^{G_2} = \alpha N_{G_2}$ and then we apply Theorem 1 to obtain the thesis. In particular we use

$$\Delta = lcm\left(N_{G_1}, m, \frac{P_{G_2} \cdot N_{G_1}}{GCD(P_{G_2} \cdot N_{G_1}, P_{G_1})}, \frac{N_{G_2} \cdot m}{GCD(N_{G_2} \cdot m, |k|)}\right)$$

and

$$\alpha = \left(\frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}\right) \cdot \frac{P_{G_2}}{N_{G_2}}$$

such that, for each i, if $\exists j, k : G'(i) = \bigcup_{r=0}^{k} G_2(j+r)$, then $G'(i+\Delta) = \bigcup_{r=0}^{k} G_2(j+r+\alpha N_{G_2})$.

Given an arbitrary granule $G'(i)$, we show that $G'(i+\Delta)$ is the union of granules that can be obtained by adding αN_{G_2} to the index of each granule of G_2 contained in $G'(i)$. Note that $i+\Delta \in \mathcal{L}_{G'}$ since G' is full-integer labeled. In order to show that this is correct we consider the way granules of G' are constructed by definition of altering-tick. More precisely, we compute the difference between the label $b'_{i+\Delta}$ of the first granule of G_2 included in $G'(i+\Delta)$ and the label b'_i of the first granule of G_2 included in $G'(i)$; we show that this difference is equal to the difference between the label $t'_{i+\Delta}$ of the last granule of G_2 included in $G'(i+\Delta)$ and the label t'_i of the last granule of G_2 included in $G'(i)$. This fact together with the consideration that G_2 is a full-integer labeled granularity, leads to the conclusion that $G'(i)$ and $G'(i+\Delta)$ have the same number of granules. It is then clear that the above computed label differences are also equal to the difference between the label of an arbitrary n-th granule of G_2 included in $G'(i+\Delta)$ and the label of the n-th granule of G_2 included in $G'(i)$. If this difference is $b'_{i+\Delta} - b'_i$, then we have: if $\exists j, k : G'(i) = \bigcup_{r=0}^{k} G_2(j+r)$, then $G'(i+\Delta) = \bigcup_{r=0}^{k} G_2\left(j+r+\left(b'_{i+\Delta} - b'_i\right)\right)$. By showing that $b'_{i+\Delta} - b'_i$ is a multiple of N_{G_2} the thesis follows.

Proof details

Assume $G_1(i) = \bigcup_{j=b_i}^{t_i} G_2(j)$ and $G_1(i+\Delta) = \bigcup_{j=b_{i+\Delta}}^{t_{i+\Delta}} G_2(j)$. We need to compute $b'_{i+\Delta} - b'_i$. From the definition of the the altering-tick operation:

$$b'_i = \begin{cases} b_i + \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) k & \text{if } i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l, \\ \\ b_i + \left(\left\lfloor \frac{i-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{7}$$

and

$$b'_{i+\Delta} = \begin{cases} b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) k & \text{if } i+\Delta = \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l, \\ \\ b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{8}$$

Note that if $i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l$, then $i + \Delta = \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l$. Indeed, $\left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l = \left(\left\lfloor \frac{i-l}{m} + \frac{\Delta}{m} \right\rfloor\right) m + l$ and, since Δ is a multiple of m, then $\left(\left\lfloor \frac{i-l}{m} + \frac{\Delta}{m} \right\rfloor\right) m + l = \left(\frac{\Delta}{m} + \left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l = \Delta + \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l$.

Hence, to compute $b'_{i+\Delta} - b'_i$ we should consider two cases:

$$b'_{i+\Delta} - b'_i = \begin{cases} b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) k - b_i - \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) k & \text{if } i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l \\ \\ b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor + 1\right) k - b_i - \left(\left\lfloor \frac{i-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{9}$$

In both cases (again considering the fact that Δ is a multiple of m):

$$b'_{i+\Delta} - b'_i = (b_{i+\Delta} - b_i) + \frac{\Delta \cdot k}{m} \tag{10}$$

We are left to compute $b_{i+\Delta} - b_i$, i.e., the distance in terms of granules of G_2, between $G_2(b_i)$ and $G_2(b_{i+\Delta})$. Since, by hypothesis, $G_1(i) = \bigcup_{j=b_i}^{t_i} G_2(j)$ and $G_1(i+\Delta) = \bigcup_{j=b_{i+\Delta}}^{t_{i+\Delta}} G_2(j)$, then the first granule of $\perp$ making $G_2(b_i)$ and the first granule of $\perp$ making $G_1(i)$ is the same granule. The same can be observed for the first granule of $\perp$ making $G_2(b_{i+\Delta})$ and the first granule of $\perp$ making $G_1(i+\Delta)$. More formally:

$$min \lfloor b_i \rfloor^{G_2} = min \lfloor i \rfloor^{G_1}$$

and

$$min \lfloor b_{i+\Delta} \rfloor^{G_2} = min \lfloor i+\Delta \rfloor^{G_1}$$

Hence, we have:

$$min \lfloor b_{i+\Delta} \rfloor^{G_2} - min \lfloor b_i \rfloor^{G_2} = min \lfloor i+\Delta \rfloor^{G_1} - min \lfloor i \rfloor^{G_1} \tag{11}$$

We have shown that the difference between the index of the first granule of $\perp$ making $G_2(b_{i+\Delta})$ and the index of the first granule of $\perp$ making $G_2(b_i)$ is equal to the difference between the index of the first granule of $\perp$ making $G_1(i+\Delta)$ and the index of the first granule of $\perp$ making $G_1(i)$. Then, we need to compute the difference between the index of the first granule of $\perp$ making $G_1(i+\Delta)$ and the index of the first granule of $\perp$ making $G_1(i)$. Since $\perp \trianglelefteq G_1$ and Δ is a multiple of N_{G_1}, for each i, if $\exists j, \tau : G_1(i) = \bigcup_{r=0}^{\tau} \perp(j+r)$, then $G_1(i+\Delta) = \bigcup_{r=0}^{\tau} \bot(j + \frac{\Delta \cdot P_{G_1}}{N_{G_1}})$. Hence, this difference has value $\frac{\Delta \cdot P_{G_1}}{N_{G_1}}$, and for what shown above this is also the value of the difference between the index of the first granule of $\bot$ making $G_2(b_{i+\Delta})$ and the index of the first granule of $\bot$ making $G_2(b_i)$. Then, since $\bot \trianglelefteq G_2$ with period length P_{G_2} and since $\frac{\Delta \cdot P_{G_1}}{N_{G_1}}$ is a multiple of P_{G_2}, we have that, if:

$$\bot(j) \subseteq G_2(i)$$

then:

$$\bot(j + \frac{\Delta \cdot P_{G_1}}{N_{G_1}}) \subseteq G_2(i + \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}})$$

Thus, $b_{i+\Delta} - b_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}}$.

Reconsidering 10:

$$b'_{i+\Delta} - b'_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}.$$

Analogously we can compute $t'_{i+\Delta} - t'_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}$.

Thus, $b'_{i+\Delta} - b'_i = t'_{i+\Delta} - t'_i$; hence $t_{i+\Delta} - b_{i+\Delta} = t_i - b_i$. Since G_2 is a full integer labeled granularity, then $G'(i)$ and $G'(i+\Delta)$ are formed by the same number of granules.

Since we now know $G'(i+\Delta) = \bigcup_{j=b'_{i+\Delta}}^{t'_{i+\Delta}} G_2(j) = \bigcup_{j=b'_i}^{t'_i} G_2(j + (b'_{i+\Delta} - b'_i))$ and $(b'_{i+\Delta} - b'_i)$ is a multiple of N_{G_2}, we have $G_2 \trianglelefteq G'$, $P_{G'}^{G_2} = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}}$ and $\bot \trianglelefteq G_2$. Hence, all the hypothesis of Theorem 1 hold, and its application leads the thesis of this proposition.

A.3.2 Part 2

Since G_2 partitions G' (see table 2.2 of (?)), then (1) $\lceil l_{G_2} \rceil_{G_2}^{G'}$ is always defined and (2) $min(\{n \in \mathbb{N}^+ | \exists i \in \mathcal{L}_{G_2} s.t. \bot(n) \subseteq G_2(i)\}) = min(\{m \in \mathbb{N}^+ | \exists j \in \mathcal{L}_{G'} s.t. \bot(m) \subseteq G'(j)\})$. Therefore $l_{G'}$ is the label of the granule of G' that covers the granule of G_2 labeled with l_{G_2}; by definition of $\lceil \cdot \rceil$ operation, $l_{G'} = \lceil l_{G_2} \rceil_{G_2}^{G'}$.

A.4 Proof of Proposition 3

A.4.1 Part 2

By definition of the *Shift* operation, $G'(i) = G(i-m)$. Hence $G'(l_G+m) = G(l_G+m-m) = G(l_G)$.

A.5 Proof of Proposition 4

A.5.1 Part 1

The thesis will follow from the application of Theorem 1. Indeed, we know that $\bot \trianglelefteq G_2$ and we show that $G_2 \trianglelefteq G'$ with $P_{G'}^{G_2}$ multiple of N_{G_2}. For this we need to identify Δ and α s.t., for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j \in s(i)} G_2(j)$, then $G'(i+\Delta) = \bigcup_{j \in s(i)} G_2(j + \alpha N_{G_2})$.

Consider an arbitrary $i \in \mathbb{N}$ and $\Delta = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$. By definition of the combining operation, we have $G'(i) = \bigcup_{j \in s(i)} G_2(j)$ and $G'(i+\Delta) = \bigcup_{j \in s(i+\Delta)} G_2(j)$ with

$$s(i) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$$

and

$$s(i+\Delta) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i+\Delta)\}.$$

We now show that $s(i+\Delta)$ is composed by all and only the elements of $s(i)$ when the quantity $\Delta' = \frac{lcm(P_{G_1}, P_{G_2})N_{G_2}}{P_{G_2}}$ is added. For this purpose we need:

$$\forall j \in s(i)\ \exists (j+\Delta') \in s(i+\Delta) \tag{12}$$

and

$$\forall \left(j+\Delta'\right) \in s(i+\Delta)\ \exists j \in s(i) \tag{13}$$

About 12, note that if $j \in s(i)$, then $G_2(j) \subseteq G_1(i)$. Since $\perp \trianglelefteq G_2$, if

$$G_2(j) = \bigcup_{r=0}^{k} \perp(j_r)$$

then

$$G_2\left(j+\Delta'\right) = \bigcup_{r=0}^{k} \perp(j_r + lcm(P_{G_1}, P_{G_2})) \tag{14}$$

Since $G_1(i) \supseteq G_2(j) = \bigcup_{r=0}^{k} \perp(j_r)$, and since $\perp \trianglelefteq G_1$, then

$$G_1(j+\Delta) \supseteq \bigcup_{r=0}^{k} \perp(j_r + lcm(P_{G_1}, P_{G_2})) \tag{15}$$

From 14 and 15 we derive $G_1(i+\Delta) \supseteq G_2(j+\Delta')$, and hence $(j+\Delta') \in s(i+\Delta)$. Analogously can be proved the validity of 13; Hence, for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j \in s(i)} G_2(j)$, then $G'(i+\Delta) = \bigcup_{j\in s(i)} G_2(j+\Delta')$. Hence, considering the fact that $G_2 \unlhd G'$, we can conclude $G_2 \trianglelefteq G'$. Finally, since $P_{G'}^{G_2}$ is a multiple of N_{G_2}, by Theorem 1 we obtain the thesis.

A.5.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \widetilde{s}(i) \neq \emptyset\}$$

where $\forall i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}\ \widetilde{s}(i) = \{j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$;

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that: (1) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that exists $k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$ and since G' is derived by the *Combine* operation, then $\exists q \in \mathcal{L}_{G_2} | G_2(q) \subseteq G_1(k)$. By definition of the *Combine* operation $G'(k) = \bigcup_{j \in s(k)} G_2(j)$; since $q \in s(k)$, then $G_2(q) \subseteq G'(k)$. Hence (a) $\exists q \in \mathcal{L}_{G_2} | G_2(q) \subseteq G'(k)$.

Moreover, since $k \notin \widetilde{\mathcal{L}}_{G'}$, then $\widetilde{s}(k) = \emptyset$; therefore $\nexists \mathbf{j} \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(k)$. By definition of the *Combine* operation it is easily seen that $G' \preceq G_1$. Using this and the previous formula, we derive that (b) $\nexists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G'(k)$.

From (a) and (b) it follows that $\exists q \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(q) \subseteq G'(k)$. We show that this leads to a contradiction.

Since $q \notin \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ and labels of $\hat{\mathcal{L}}_{G_2}^{P_{G'}}$ are contiguous (i.e., $\nexists i \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ s.t. $min(\hat{\mathcal{L}}_{G_2}^{P_{G'}}) < i < max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$), then $q < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ or $q > max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$. We consider the first case, the proof for the second is analogous.

If $q < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ then $max(\lfloor q \rfloor^{G_2}) < 1$ (otherwise $q \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$).

Let be $\alpha = min(\lfloor min(\hat{\mathcal{L}}_{G'}) \rfloor^{G'})$. Since $k \in \hat{\mathcal{L}}_{G'}$, then $\alpha \leq \lfloor k \rfloor^{G'}$.

If $\alpha \geq 1$, then $G'(k) \cap G_2(q) = \emptyset$ contradicting $G'(k) \supseteq G_2(q)$.

If $\alpha < 1$, then $G'(l_{G'}) \supseteq \perp(0)$ and we show that $l_{G'} \in \widetilde{\mathcal{L}}_{G'}$. Indeed, by definition of *Combine*, $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G'(L_{G'})$. Since $G' \preceq G_1$ we also have $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(L_{G'})$; hence $j \in \widetilde{s}(l_{G'})$ and then $l_{G'} \in \widetilde{\mathcal{L}}_{G'}$.

Since $0 \in G'(l_{G'})$ and $max(\lfloor q \rfloor^{G_2}) \leq 0$, then $max(\lfloor q \rfloor^{G_2}) < \alpha$ (otherwise $G_2(q) \subseteq G'(l_{G'})$). Therefore, since $min(\lfloor k \rfloor^{G'}) \geq \alpha$, then $\lfloor q \rfloor^{G_2} \cap \lfloor l_{G'} \rfloor^{G'} = \emptyset$, in contradiction with $G_2(q) \subseteq G'(k)$.

(2) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$, by definition of $\widetilde{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ and $\widetilde{s}(k) \neq \emptyset$; Therefore, by definition of $\widetilde{s}$, $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(k)$.

Since $j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$, by definition of $\hat{\mathcal{L}}$, $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_2} = j$. Since $G_2(j) \subseteq G_1(k)$, then $\lceil h \rceil^{G_1} = k$. By definition of the *combine* operation, $\lceil h \rceil^{G'} = k$. Moreover, since $0 < h \leq P_{G'}$, by definition of $\hat{\mathcal{L}}$, $\lceil h \rceil^{G'} = k \in \hat{\mathcal{L}}_{G'}$, contradicting the hypothesis.

A.6 Proof of Proposition 5

A.6.1 Part 1

The thesis will follow from the application of Theorem 1. Indeed, we show that $G_1 \trianglelefteq G'$ with $P_{G'}^{G_1}$ multiple of N_{G_1}. For this we need to identify Δ and α s.t., for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j \in s(i)} G_1(j)$, then $G'(i+\Delta) = \bigcup_{j \in s(i)} G_1(j + \alpha N_{G_1})$. Let $\Delta = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$. By definition of anchored grouping, $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ and $G'(i+\Delta) = \bigcup_{j=i+\Delta}^{(i+\Delta)'-1} G_1(j)$ where i' is the first label of G_2 after i and $(i+\Delta)'$ is the first label of G_2 after $i+\Delta$. By periodicity of G_2, (and since Δ is a multiple of N_{G_2}) the difference between the label of the granule following $G_2(i+\Delta)$ and the label of the granule following $G_2(i)$ is Δ. More formally, $(i+\Delta)' - i' = \Delta$, hence $(i+\Delta)' = i' + \Delta$. Then, for each i, if $G'(i) = \bigcup_{j=i}^{k} G_1(j)$, then $G'(i+\Delta) = \bigcup_{j=i+\Delta}^{i'+\Delta-1} G_1(j) = \bigcup_{j=i}^{i'-1} G_1(j+\Delta)$. By this result and considering $G_1 \trianglelefteq G'$, we conclude $G_1 \trianglelefteq G'$ with $P_{G'}^{G_1} = \Delta$. Note that by Proposition 9, $N_{G_1} = \frac{P_{G_1} \cdot N_{G_2}}{P_{G_2}}$, hence $P_{G'}^{G_1}$ is a multiple of Δ. Then, by Theorem 1, we have the thesis.

A.6.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \begin{cases} \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{if } l_{G_2} = l_{G_1}, \\ \{l'_{G_2}\} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{otherwise,} \end{cases}$$

where l'_{G_2} is the greatest among the labels of $\mathcal{L}_{G_2}$ that are smaller than l_{G_2}. We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\hat{\mathcal{L}}_{G'} \subseteq \widetilde{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Then, since $k \in \widetilde{\mathcal{L}}_{G'}$, then $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ or $k = l'_{G_2}$.

If $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$, then, by definition of $\hat{\mathcal{L}}_{G_2}^{P_{G'}}$, $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_2} = k$. By definition of *Anchored-group*, $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ where k' is the first label of G_2 after k. Therefore $G'(k) \supseteq G_1(k)$. Since G_2 is a labeled aligned subgranularity of G_1 and since $k \in \mathcal{L}_{G_2}$, then $k \in \mathcal{L}_{G_1}$ and $G_1(k) = G_2(k)$. Hence $G'(k) \supseteq G_2(k)$. It follows that $\lceil h \rceil^{G'} = k$ and therefore, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

If $k = l'_{G_2}$, then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $l_{G_2} \neq l_{G_1}$. Therefore, since G_2 is a labeled aligned subgranularity of G_1 $l'_{G_2} < l_{G_1} < l_{G_2}$; then $\exists h$ with $0 < h < min(\lfloor l_{G_2} \rfloor^{G_2})$ s.t. $\lceil h \rceil^{G_1} = l_{G_1}$. Since, by definition of *Anchored-group*, $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$ and since $l'_{G_2} < l_{G_1} < l_{G_2}$, then $G'(l'_{G_2}) \supseteq G_1(l_{G_1})$. Hence $\lceil h \rceil^{G'} = l'_{G_2}$ and therefore, by definition of $\hat{\mathcal{L}}$, $l'_{G_2} = k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. If $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $k \in \widetilde{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

If $k \notin \hat{\mathcal{L}}_{G_2}^{P_{G'}}$, since $\nexists q \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ s.t. $min(\hat{\mathcal{L}}_{G_2}^{P_{G'}}) \leq q \leq max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$, then $k > max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ or $k < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$.

If $k > max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ then, by definition of $\hat{\mathcal{L}}$, $min(\lfloor k \rfloor^{G_2}) > P_{G'}$. Since G_2 is a labeled aligned subgranularity of G_1 then $G_2(k) = G_1(k)$ and hence $min(\lfloor k \rfloor^{G_1}) > P_{G'}$. Since $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ then $min(\lfloor k \rfloor^{G'}) > P_{G'}$ in contrast with the hypothesis $k \in \hat{\mathcal{L}}_{G'}$.

If $k < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ then, by definition of l'_{G_2}, $k < l'_{G_2}$ or $k = l'_{G_2}$.

If $k < l'_{G_2}$ then, let k' be the next label of G_2 after k. Since $k < l'_{G_2}$ then, by definition l'_{G_2}, $k' \leq l'_{G_2}$. By definition of l'_{G_2} then $max(\lfloor l'_{G_2} \rfloor^{G_2}) \leq 0$. Since G_2 is a labeled aligned subgranularity of G_1 then $G_1(l'_{G_2}) = G_2(l'_{G_2})$; therefore $max(\lfloor l'_{G_2} \rfloor^{G_1}) \leq 0$. Since $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ and $k' \leq l'_{G_2}$, follows that $max(\lfloor k \rfloor^{G'}) \leq 0$ in contrast with the hypothesis $k \in \hat{\mathcal{L}}_{G'}$.

Finally if $k = l'_{G_2}$ then $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$. Since $k = l'_{G_2} \in \hat{\mathcal{L}}_{G'}$ then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = l'_{G_2}$. Since G' is the composition of granules of G_1, $\lceil h \rceil^{G_1}$ is defined. Let $q = \lceil h \rceil^{G_1}$. By definition of $\hat{\mathcal{L}}$, $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ and therefore $q \geq l_{G_1}$. Since, by definition of *Anchored-group*, G' is the composition of granules of G_1 and since $\lceil h \rceil^{G'} = l'_{G_2}$ and $\lceil h \rceil^{G_1} = q$, then $G_1(q) \subseteq G'(l'_{G_2})$. Therefore since $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$ then $q < l_{G_2}$. It follows that $l_{G_1} \leq q < l_{G_2}$ and hence $l_{G_1} \neq l_{G_2}$. By definition of $\widetilde{\mathcal{L}}_{G'}$, $l'_{G_2} = k \in \widetilde{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

A.7 Selecting operations

The selecting operations have a common part in the proof for the computation of the period length and the period label distance.

Let be $\Gamma = \frac{lcm(P_{G_1},P_{G_2})N_{G_1}}{P_{G_1}}$. The proof is divided into two steps: first we show that for each select operation if $i \in \mathcal{L}_{G'}$ then $i + \Gamma \in \mathcal{L}_{G'}$ (details for *Select-down*, *Select-up* and *Select-by-intersect* operations can be found below). The second step is the application of Theorem 1. Indeed, for each *Select* operation, the following holds: $\forall i \in \mathcal{L}_{G'}$ $G'(i) = G_1(i)$; this implies $G_1 \trianglelefteq G'$. From step 1 follows that $i + \Gamma \in \mathcal{L}_{G'}$, hence $G'(i + \Gamma) = G_1(i + \Gamma)$. By this result and considering $G_1 \trianglelefteq G'$, we conclude that $G_1 \bar{\trianglelefteq} G'$ with $P^{G_1}_{G'} = \Gamma$ which is a multiple of N_{G_1} by definition. Then, by Theorem 1 we have the thesis.

A.8 Proof of Proposition 6

A.8.1 Part 1

See Section A.7.

We prove that if $\lambda \in \mathcal{L}_{G'}$ then $\lambda' = \lambda + \Gamma \in \mathcal{L}_{G'}$.

By definition of the `select-down` operation, if $\lambda \in \mathcal{L}_{G'}$ then $\exists i \in \mathcal{L}_{G_2}$ s.t. $\lambda \in \Delta^l_k\left(S(i)\right)$ where $S(i)$ is an ordered set defined as follows: $S(i) = \{j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i)\}$. In order to prove the thesis we need to show that $\exists i' \in \mathcal{L}_{G_2} | \lambda' \in \Delta^l_k(S(i'))$. Consider $i' = i + \frac{lcm(P_{G_1}P_{G_2})N_{G_2}}{P_{G_2}}$ we will note that $i' \in \mathcal{L}_{G_2}$ (this is trivially derived from the periodicity of G_2). To prove that $\lambda' \in \Delta^l_k(S(i'))$ we show that $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements.

Indeed note that from periodicity of G_1, $\forall j \in S(i)$ if:

$$G_1(j) = \bigcup_{r=0}^{\tau_j} \perp(j_r) \tag{16}$$

then:

$$G_1\left(j'\right) = \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1}P_{G_2})) \tag{17}$$

Since $j \in S(i)$, $G_1(j) \subseteq G_2(i)$ then, from (16), $G_2(i) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r)$. Moreover, from periodicity of G_2:

$$G_2\left(i'\right) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1}P_{G_2})) \tag{18}$$

Since (17) and (18), $G_2\left(i'\right) \supseteq G_1\left(j'\right)$; hence $\forall j \in S(i), j' = (j + \Gamma) \in S(i')$. Analogously we can prove that $\forall j' \in S(i'), j = (j' - \Gamma) \in S(i)$.

Thus $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements; therefore if $j \in S(i)$ has position n in $S(i)$, so $j' \in S(i')$ has position n in $S(i')$. Hence it is trivial that if λ has position between k and $k + l - 1$ in $S(i)$, then λ' has position between k and $k + l - 1$ in $S(i')$. Hence if $\lambda \in \mathcal{L}_{G'}$, then $\lambda' \in \mathcal{L}_{G'}$.

A.8.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}} \left\{ a \in A(i) | a \in \hat{\mathcal{L}}^{P_{G'}}_{G_1} \right\};$$

where $\forall i \in \mathcal{L}_{G_2}$:

$$A(i) = \Delta^l_k \left(\{ j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i) \} \right).$$

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1)Suppose by contradiction that $\exists q \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. By definition of $\widetilde{\mathcal{L}}_{G'}$, $q \in \hat{\mathcal{L}}^{P_{G'}}_{G_1}$; therefore $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_1} = q$. Moreover, by definition of $\widetilde{\mathcal{L}}_{G'}$ and by definition of *Select-down*, $\widetilde{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$ hence $q \in \mathcal{L}_{G'}$. Since, by definition of *Select-down* $G'(q) = G_1(q)$, then $\lceil h \rceil^{G'} = q$; hence, by definition of $\hat{\mathcal{L}}$, $q \in \hat{\mathcal{L}}_{G'}$ in contradiction with hypothesis.

(2)Suppose by contradiction that $\exists q \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $q \in \hat{\mathcal{L}}_{G'}$ then, by definition of *Select-down*

$$\exists i \in \mathcal{L}_{G_2} \, s.t. \, q \in \Delta^l_k \left(\{ j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i) \} \right)$$

therefore, by definition of $A(i)$, $q \in A(i)$.

Since $q \in \hat{\mathcal{L}}_{G'}$ then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = q$. By definition of *Select-down*, $G'(q) = G_1(q)$, then $\lceil h \rceil^{G_1} = q$ and therefore $q \in \hat{\mathcal{L}}^{P_{G'}}_{G_1}$. Moreover, since $G_1(q) \subseteq G_2(i)$, then $\lceil h \rceil^{G_2} = i$ and therefore $i \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}$. Since $q \in A(i)$, $q \in \hat{\mathcal{L}}^{P_{G'}}_{G_1}$ and $i \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}$ then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $q \in \widetilde{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

A.9 Proof of Proposition 7

A.9.1 Part 1

See Section A.7. We prove that if $i \in \mathcal{L}_{G'}$ then $i + \Gamma \in \mathcal{L}_{G'}$. From the periodicity of G_1, $i + \Gamma \in \mathcal{L}_{G_1}$ (this is trivially derived from the periodicity of G_1). Hence we only need to show that $\exists j' \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i + \Gamma)$. Since $i \in \mathcal{L}_{G'}$ then $\exists j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)$.

From the periodicity of G_2, if:

$$G_2(j) = \bigcup_{r=0}^{\tau_j} \perp(j_r) \tag{19}$$

then:

$$G_2 \left(j + \frac{lcm(P_{G_1} P_{G_2}) N_{G_2}}{P_{G_2}} \right) = \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1} P_{G_2})) \tag{20}$$

Moreover, from the (19) and since $G_1(i) \supseteq G_2(j)$:

$$G_1(i) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r)$$

From the periodicity of G_1:

$$G_1(i+\Gamma) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1}P_{G_2})) \tag{21}$$

From (20) and (21) follows that $G_1(i+\Gamma) \supseteq G_2\left(j + \frac{lcm(P_{G_1}P_{G_2})N_{G_2}}{P_{G_2}}\right)$, that is the thesis.

A.9.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \exists j \in \mathcal{L}_{G_2}\, s.t.\, \emptyset \neq G_2(j) \subseteq G_1(i)\};$$

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G_2}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$, then $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$; therefore $\exists h$ with $0 < h \leq P_{G'}$ s. t. $\lceil h \rceil^{G_1} = k$. Moreover, by definition of $\widetilde{\mathcal{L}}_{G'}$ and by definition of *Select-down*, $\widetilde{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$ hence $q \in \mathcal{L}_{G'}$. Since, by definition of *Select-up*, $G'(k) = G_1(k)$, then $\lceil h \rceil^{G'} = k$. Hence, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$, then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Since, by definition of *Select-up*, $G'(k) = G_1(k)$, then $\lceil h \rceil^{G_1} = k$; Therefore, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$. Moreover, since $k \in \hat{\mathcal{L}}_{G'}$ and $\hat{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$, by definition of the *Select-up* operation, then $\exists j \in \mathcal{L}_{G_2}$ s.t. $\emptyset \neq G_2(j) \subseteq G_1(k)$. Hence by definition of $\widetilde{\mathcal{L}}_{G'}$, $k \in \widetilde{\mathcal{L}}_{G'}$, in contradiction with hypothesis.

A.10 Proof of Proposition 8

A.10.1 Part 1

See Section A.7. We prove that if $\lambda \in \mathcal{L}_{G'}$, then $\lambda' = \lambda + \Gamma \in \mathcal{L}_{G'}$.

By definition of the *select-by-intersect* operation, if $\lambda \in \mathcal{L}_{G'}$, then $\exists i \in \mathcal{L}_{G_2} : \lambda \in \Delta_k^l(S(i))$ where $S(i)$ is an ordered set defined as follows: $S(i) = \{j \in \mathcal{L}_{G_1} | G_1(j) \cap G_2(i) \neq \emptyset\}$. In order to prove the thesis we need to show that $\exists i' \in \mathcal{L}_{G_2} : \lambda' \in \Delta_k^l(S(i'))$. Consider $i' = i + \frac{lcm(P_{G_1}P_{G_2})N_{G_2}}{P_{G_2}}$ note that $i' \in \mathcal{L}_{G_2}$ (this is trivially derived from the periodicity of G_2). To prove that $\lambda' \in \Delta_k^l(S(i'))$ we show that $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements.

Indeed note that $\forall j$ if $j \in S(i)$, then $G_1(j) \cap G_2(i) \neq \emptyset$. Hence $\exists l \in \mathbb{Z} : \perp(l) \subseteq G_1(j)$ and $\perp(l) \subseteq G_2(i)$. From the periodicity of G_1, $G_1(j+\Gamma) \supseteq \perp(l + lcm(P_{G_1}P_{G_2}))$. From the periodicity of G_2, $G_2(i') \supseteq \perp(l + lcm(P_{G_1}P_{G_2}))$. So $G_1(j+\Gamma) \cap G_2(i') \neq \emptyset$, therefore $\forall j \in S(i), (j+\Gamma) \in S(i')$.

Analogously we can prove that $\forall j' \in S(i'), (j' - \Gamma) \in S(i)$. Hence $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements. Therefore, if $j \in S(i)$ has position n in $S(i)$, then $j + \Gamma \in S(i')$ has position n in $S(i')$; hence if j has position between k and $k+l-1$ in $S(i)$, then also $j+\Gamma$ has position between k and $k+l-1$ in $S(i')$ and so $j+\Gamma \in \mathcal{L}_{G'}$.

A.10.2 Part 2

The proof is analogous to the ones of Proposition 6.

A.11 Set Operations

A.11.1 Proof of Proposition 9

Given the periodical granularities H and G with G label aligned subgranularity of H, we prove that $\frac{N_G}{P_G} = \frac{N_H}{P_H}$. The thesis is proved by considering the common period length of H and G i.e. $P_c = lcm(P_G, P_H)$.

Let N'_G be the difference between the label of the i^{th} granule of one period of G and the label of the i^{th} granule of the next period, considering P_c as the period length of G. Analogously N'_H is defined.

By periodicity of G, if $G(i) = \bigcup_{r=0}^{k} \perp(i_r)$ then $G(i + N'_G) = \bigcup_{r=0}^{k} \perp(i_r + P_c)$; since G is an aligned subranularity of H, $\forall i \in \mathcal{L}_H$ $H(i) = G(i) = \bigcup_{r=0}^{k} \perp(i_j)$ and, since H is periodic, $H(i + N'_H) = \bigcup_{r=0}^{k} \perp(i_j + P_c)$; from which we can easily derive that $i + N'_G = i + N'_H$, hence $N'_G = N'_H$.

From the definition of P_c, $\exists \alpha, \beta \in \mathbb{N}$ s. t. $\alpha P_H = \beta P_G$. Moreover, since $N'_H = N'_G$, then $\alpha N_H = \beta N_G$. Therefore $\frac{P_H}{N_H} = \frac{P_G}{N_G}$.

A.11.2 Property used in the proofs for set operations

Let Γ_1 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$ and Γ_2 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$. Since G_1 and G_2 are aligned subgranularity of a certain granularity H, from Proposition 9 we can easily derive that $\Gamma_1 = \Gamma_2$.

A.12 Proof of Proposition 10

A.12.1 Part 1

Union. Let Γ_1 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$ and Γ_2 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$. The thesis will be proved by showing that $\forall i \in \mathcal{L}_{G'}$ if, $G'(i) = \bigcup_{r=0}^{k} \perp(i_r)$, then $G'(i + \Delta) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$ with $\Delta = \Gamma_1 = \Gamma_2$. Since $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cup \mathcal{L}_{G_2}$, two cases will be considered:

- $\forall i \in \mathcal{L}_{G_1}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_1, $G_1(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence $G'(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$.
- $\forall i \in \mathcal{L}_{G_2} - \mathcal{L}_{G_1}$ $G'(i) = G_2(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_2, $G_2(i + \Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence $G'(i + \Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$.

Since $\Gamma_1 = \Gamma_2$, then $\forall i \in \mathcal{L}_{G'}$ if $G'(i) = \bigcup_{r=0}^{k} \perp(i_r)$, then $G'(i + \Gamma_1) = G'(i + \Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$. Hence, by definition of $\trianglelefteq$, we have the thesis.

Intersect. $\forall i \in \mathcal{L}_{G'} = \mathcal{L}_{G_1} \cap \mathcal{L}_{G_2}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_1 and G_2, $i + \Gamma_1 \in \mathcal{L}_{G_1}$ e $i + \Gamma_2 \in \mathcal{L}_{G_2}$; since $\Gamma_1 = \Gamma_2$, then $i + \Gamma_1 \in \mathcal{L}_{G'}$. Moreover $G'(i + \Gamma_1) = G_1(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence, by the definition of $\trianglelefteq$, we have the thesis.

Difference. $\forall i \in \mathcal{L}_{G'} = \mathcal{L}_{G_1} - \mathcal{L}_{G_2}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. Since $i \in \mathcal{L}_{G_1}$ from the periodicity of G_1 $i + \Gamma_1 \in \mathcal{L}_{G_1}$. Since $i \notin \mathcal{L}_{G_2}$, from the periodicity of G_2, $i + \Gamma_2 \notin \mathcal{L}_{G_2}$ (if it would exists $i + \Gamma_2 \in \mathcal{L}_{G_2}$, from periodicity of G_2 would exists $i \in \mathcal{L}_{G_2}$ that is not possible for hypothesis). Hence $i + \Gamma_1 \in \mathcal{L}_{G'}$. Moreover $G'(i + \Gamma_1) = G_1(i + \Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence, by the definition of $\trianglelefteq$, we have the thesis.

A.12.2 Part 2

Let $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G_1}^{P_{G'}} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}$.

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$ then $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ or $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$. Suppose that $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ (the proof is analogous if $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$). Since $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$, then $\exists\ 0 < h < P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Since, by definition of the *Union* operation $G'(k) = G_1(k)$, then $\lceil h \rceil^{G'} = k$. Hence, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$, then, by definition of $\hat{\mathcal{L}}$, $\exists\ 0 < h < P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Moreover, by definition of the *Union* operation, $k \in \mathcal{L}_{G_1}$ or $k \in \mathcal{L}_{G_2}$. Suppose that $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ (the proof is analogous if $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$). By definition of the *Union* operation, $G'(k) = G_1(k)$ therefore $\lceil h \rceil^{G_1} = k$ and so, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$. Hence, by definition of $\widetilde{\mathcal{L}}$, $k \in \widetilde{\mathcal{L}}_{G'}$ in contradiction with the hypothesis.

Supporting Temporal Reasoning by Mapping Calendar Expressions to Minimal Periodic Sets

Claudio Bettini   Sergio Mascetti
Dipartimento di Informatica e Comunicazione,
Università di Milano
{bettini,mascetti}@dico.unimi.it

X. Sean Wang
Department of Computer Science,
University of Vermont, Burlington, VT, USA
Sean.Wang@uvm.edu

Abstract

In the recent years several research efforts have focused on the concept of time granularity and its applications. A first stream of research investigated the mathematical models behind the notion of granularity and the algorithms to manage temporal data based on those models. A second stream of research investigated symbolic formalisms providing a set of algebraic operators to define granularities in a compact and compositional way. However, only very limited manipulation algorithms have been proposed to operate directly on the algebraic representation making it unsuitable to use the symbolic formalisms in applications that need manipulation of granularities.

This paper aims at filling the gap between the results from these two streams of research, by providing an efficient conversion from the algebraic representation to the equivalent low-level representation based on the mathematical models. In addition, the conversion returns a minimal representation in terms of period length. Our results have a major practical impact: users can more easily define arbitrary granularities in terms of algebraic operators, and then access granularity reasoning and other services operating efficiently on the equivalent, minimal low-level representation. As an example, we illustrate the application to temporal constraint reasoning with multiple granularities.

From a technical point of view, we propose an hybrid algorithm that interleaves the conversion of calendar subexpressions into periodical sets with the minimization of the period length. The algorithm returns set-based granularity representations having minimal period length, which is the most relevant parameter for the performance of the considered reasoning services. Extensive experimental work supports the techniques used in the algorithm, and shows the efficiency and effectiveness of the algorithm.

1 Introduction

According to a 2006 research by Oxford University Press, the word *time* has been found to be the most common noun in the English language, considering diverse sources on the Internet including newspapers, journals, fictions and weblogs. What is somehow surprising is that among the 25 most common nouns we find time granularities like *day*, *week*, *month* and *year*. We are pretty sure that many other time granularities like *business day*, *quarter*, *semester*, etc. would be found to be quite frequently used in natural languages. However, the way computer applications deal with these concepts is still very naive and mostly hidden in program code and/or based on limited and sometimes imprecise calendar support. Temporal representation and reasoning has been for a long time an AI research topic aimed at providing a formal framework for common sense reasoning, natural language understanding, planning, diagnosis and many other complex tasks involving time data management. Despite the many relevant contributions, time granularity representation and reasoning support has very often been ignored or over-simplified. In the very active area of temporal constraint satisfaction, most proposals implicitly assumed that adding support for granularity was a trivial extension. Only quite recently it was recognized that this is not the case and specific techniques were proposed (?). Even the intuitively simple task of deciding whether a specific instant is part of a time granularity can be tricky when arbitrary user-defined granularities like e.g., *banking days*, or *academic semesters* are considered.

Granularities and periodic patterns in terms of granularities are playing a role even in emerging application areas like inter-organizational workflows and personal information management (PIM). For example, inter-organizational workflows need to model and monitor constraints like: *Event2 should occur no later than two business days after the occurrence of Event1.* In the context of PIM, current calendar applications, even on mobile devices, allow the user to specify quite involved periodical patterns for the recurrence of events. For example, it is possible to schedule an event every last Saturday of every two months. The complexity of the supported patterns has been increasing in the last years, and the current simple interfaces are showing their limits. They are essentially based on a combination of recurrences based on one or two granularities taken from a fixed set (days, weeks, months, and years). We foresee the possibility for significant extensions of these applications by specifying recurrences over user-defined granularities. For example, the user may define (or upload from a granularity library) the granularity corresponding to the academic semester of the school he is teaching at, and set the date of the finals as the last Monday of each semester. A bank may want to define its *banking days* granularity and some of the bank policies may then be formalized as recurrences in terms of that granularity. Automatically generated appointments from these policies may appear on the devices of bank employees involved in specific procedures. We also foresee the need to show a user preferred view of the calendar. With current standard applications the user has a choice between a business-day limited view and a complete view, but why not enabling a view based on the users's *consulting-days*, for example? A new perspective in the use of mobile devices may also result from considering the time span in which activities are supposed to be executed (expressed in arbitrary granularities), and having software agents on board to alert about constraints that may be violated, even based on contextual information like the user location or traffic conditions. This scenario highlights three main requirements: a) a sufficiently expressive formal model for time granularity, b) a convenient way to define new time granularities, and c) efficient reasoning tools over time granularities.

Consider a). In the last decade significant efforts have been made to provide formal models for the notion of time granularity and to devise algorithms to manage temporal data based on those models. In addition to *logical* approaches (?, ?), a framework based on periodic-set representations has been extensively studied (?), and more recently an approach based on strings and automata was introduced (?, ?). We are mostly interested in the last two approaches because they support the effective computation of basic operations on time granularities. In both cases the representation of granularities can be considered as a *low-level* one, with a rather involved specification in terms of the instants of the time domain.

Consider requirement b) above. Users may have a hard time in defining granularities in formalisms based on low-level representations, and to interpret the output of operations. It is clearly unreasonable to ask users to specify granularities by linear equations or other mathematical formalisms that operate directly in terms of instants or of granules of a fixed time granularity. Hence, a second stream of research investigated more *high-level* symbolic formalisms providing a set of algebraic operators to define granularities in a compact and compositional way. The efforts on this task started even before the research on formal models for granularity (?, ?) and continued as a parallel stream of research (?, ?, ?, ?, ?).

Finally, let us consider requirement c) above. Several inferencing operations have been defined on low-level representations, including equivalence, inclusion between granules in different granularities, and even complex inferencing services like constraint propagation (?). Even for simple operations no general method is available operating directly on the high level representation. Indeed, in some cases, the proposed methods cannot exploit the structure of the expression and require the enumeration of granules, which may be very inefficient. This is the case, for example, of the granule conversion methods presented by Ning e at. (?). Moreover, we are not aware of any method to perform other operations, such as equivalence or intersection of sets of granules, directly in terms of the high level representation.

The major goal of this paper is to provide a unique framework to satisfy the requirements a), b), and c) identified above, by adding to the existing results a smart and efficient technique to convert granularity specifications from the high-level algebraic formalism to the low-level one, for which many more reasoning tools are available. In particular, in this paper we focus on the conversion from the high-level formalism called *Calendar Algebra* (?) to the low-level formalism based on periodical sets (?, ?). Among the several proposals for the high-level (algebraic) specification of granularities, the choice of Calendar Algebra has two main motivations: first, it allows the user to express a large class of granularities; For a comparison of the expressiveness of Calendar Algebra with other formalisms see (?). Second, it provides the richest set of algebraic operations that are designed to reflect the intuitive ways in which users define new granularities. A discussion on the actual usability of this tool and on how it could be enhanced by a graphical user interface can be found in Section 6.2. The choice of the low-level formalism based on periodic-sets also has two main motivations: first, an efficient implementation of all the basic operations already exists and has been extensively experimented (?); second, it is the only one currently supporting the complex operations on granularities needed for constraint satisfaction, as it will be illustrated in more detail in Section 6.1.

The technical contribution of this paper is a hybrid algorithm that interleaves the conversion of calendar subexpressions into periodical sets with a step for period minimization. A central phase of our conversion procedure is to derive, for each algebraic subexpression, the periodicity of the output set. This periodicity is used to build the periodical representation of the subexpression that can be recursively used as operand of other expressions. Given a calendar algebra expression, the algorithm returns set-based granularity representations having minimal period length. The period length is the most relevant parameter for the performance both of basic operations on granularities and of more specialized ones like the operations used by the constraint satisfaction service. Extensive experimental work reported in this paper validates the techniques used in the algorithm, by showing, among other things, that (1) even large calendar expressions can be efficiently converted, and (2) less precise conversion formulas may lead to unacceptable computation time. This latter property shows the importance of carefully and accurately designed conversion formulas. Indeed, conversion formulas may seem trivial if the length of periodicity is not a concern. In designing our conversion formulas, we made an effort to reduce the period length of the resulting granularity representation, and thus render the whole conversion process computationally efficient.

In the next section we define granularities; several interesting relationships among them are highlighted and the periodical set representation is formalized. In Section 3 we define Calendar Algebra and present its operations. In Section 4 we describe the conversion process: after the definition of the three steps necessary for the conversion, for each algebraic operation we present the formulas to perform each step. In Section 5 we discuss the period minimality issue, and we report experimental results based on a full implementation of the conversion algorithm and of its extension ensuring minimality. In Section 6 we further motivate our work by presenting a complete application scenario. Section 7 reports the related work, and Section 8 concludes the paper.

2 Formal Notions of Time Granularities

Time granularities include very common ones like hours, days, weeks, months and years, as well as the evolution and specialization of these granularities for specific contexts or applications. Trading days, banking days, and academic semesters are just few examples of specialization of granularities that have become quite common when describing policies and constraints.

2.1 Time Granularities

A comprehensive formal study of time granularities and their relationships can be found in (?). In this paper, we only introduce notions that are essential to show our results. In particular, we report here the notion of *labeled granularity* which was proposed for the specification of a calendar algebra (?, ?); we will show later how any *labeled granularity* can be reduced to a more standard notion of granularity, like the one used by Bettini et al. (?).

Granularities are defined by grouping sets of instants into *granules.* For example, each granule of the granularity `day` specifies the set of instants included in a particular day. A label is used to refer to a particular granule. The whole set of time instants is called *time domain*, and for the purpose of this paper the domain can be an arbitrary infinite set with a total order relationship, $\leq$.

Definition 1 *A* labeled granularity *G is a pair $(\mathcal{L}_G, M)$, where $\mathcal{L}_G$ is a subset of the integers, and M is a mapping from $\mathcal{L}_G$ to the subsets of the time domain such that for each pair of integers i and j in $\mathcal{L}_G$ with $i < j$, if $M(i) \neq \emptyset$ and $M(j) \neq \emptyset$, then (1) each element in $M(i)$ is less than every element of $M(j)$, and (2) for each integer k in $\mathcal{L}_G$ with $i < k < j$, $M(k) \neq \emptyset$.*

The former condition guarantees the "monotonicity" of the granularity; the latter is used to introduce the bounds (see Section 2.2).

We call $\mathcal{L}_G$ the *label set* and for each $i \in \mathcal{L}_G$ we call $G(i)$ a *granule*; if $G(i) \neq \emptyset$ we call it a *non-empty granule*. When $\mathcal{L}_G$ is exactly the integers, the granularity is called "full-integer labeled". When $\mathcal{L}_G = \mathbb{Z}^+$ we have the same notion of granularity as used in several applications, e.g., (?). For example, following this labeling schema, if we assume to map `day`(1) to the subset of the time domain corresponding to January 1, 2001, `day`(32) would be mapped to February 1, 2001, `b-day`(6) to January 8, 2001 (the sixth business day), and `month`(15) to March 2002. The generalization to arbitrary label sets has been introduced mainly to facilitate conversion operations in the algebra, however our final goal is the conversion of a labeled granularity denoted by a calendar expression into a "positive-integer labeled" one denoted by a periodic formula.

2.2 Granularity Relationships

Some interesting relationships between granularities follows. The definitions are extended from the ones presented by Bettini et al. (?) to cover the notion of labeled granularity.

Definition 2 *If G and H are labeled granularities, then G is said to* group into *H, denoted $G \trianglelefteq H$, if for each non-empty granule $H(j)$, there exists a (possibly infinite) set S of labels of G such that $H(j) = \bigcup_{i \in S} G(i)$.*

Intuitively, $G \trianglelefteq H$ means that each granule of H is a union of some granules of G. For example, `day` $\trianglelefteq$ `week` since a week is composed of 7 days and `day` $\trianglelefteq$ `b-day` since each business day is a day.

Definition 3 *If G and H are labeled granularities, then G is said to be* finer than *H, denoted $G \preceq H$, if for each granule $G(i)$, there exists a granule $H(j)$ such that $G(i) \subseteq H(j)$.*

For example `business-day` is finer than `day`, and also finer than `week`.

We also say that G *partitions* H if $G \trianglelefteq H$ and $G \preceq H$. Intuitively G partitions H if $G \trianglelefteq H$ and there are no granules of G other than those included in granules of H. For example, both `day` and `b-day` group into `b-week` (business week, i.e., the business day in a week), but `day` does not partition `b-week`, while `b-day` does.

Definition 4 *A labeled granularity G_1 is a* label-aligned subgranularity *of a labeled granularity G_2 if the label set $\mathcal{L}_{G_1}$ of G_1 is a subset of the label set $\mathcal{L}_{G_2}$ of G_2 and for each i in $\mathcal{L}_{G_1}$ such that $G_1(i) \neq \emptyset$, we have $G_1(i) = G_2(i)$.*

Intuitively, G_1 has a subset of the granules of G_2 and those granules have the same label in the two granularities.

Granularities are said to be *bounded* when $\mathcal{L}_G$ has a first or last element or when $G(i) = \emptyset$ for some $i \in \mathcal{L}_G$. We assume the existence of an unbounded bottom granularity, denoted by $\perp$ which is full-integer labeled and groups into every other granularity in the system.

There are time domains such that, given any set of granularities, it is always possible to find a bottom one; for example, it can be easily proved that this property holds for each time domain that has the same cardinality as the integers. On the other hand, the same property does not hold for other time domains (e.g. the reals). However, the assumption about the existence of the bottom granularity is still reasonable since we address problems in which granularities are defined starting from a bottom one. The definition of a *calendar* as a set of granularities that have the same bottom granularity (?) captures this idea.

2.3 Granularity Conversions

When dealing with granularities, we often need to determine the granule (if any) of a granularity H that covers a given granule z of another granularity G. For example, we may wish to find the month (an interval of the absolute time) that includes a given week (another interval of the absolute time).

This transformation is obtained with the *up* operation. Formally, for each label $z \in \mathcal{L}_G$, $\lceil z \rceil_G^H$ is undefined if $\nexists z' \in \mathcal{L}_H$ s.t. $G(z) \subseteq H(z')$; otherwise,

$\lceil z \rceil_G^H = z'$, where z' is the unique index value such that $G(z) \subseteq H(z')$. The uniqueness of z' is guaranteed by the monotonicity [1] of granularities. As an example, $\lceil z \rceil_{\texttt{second}}^{\texttt{month}}$ gives the month that includes the second z. Note that while $\lceil z \rceil_{\texttt{second}}^{\texttt{month}}$ is always defined, $\lceil z \rceil_{\texttt{week}}^{\texttt{month}}$ is undefined if week z falls between two months. Note that if $G \preceq H$, then the function $\lceil z \rceil_G^H$ is defined for each index value z. For example, since $\texttt{day} \preceq \texttt{week}$, $\lceil z \rceil_{\texttt{day}}^{\texttt{week}}$ is always defined, i.e., for each day we can find the week that contains it. The notation $\lceil z \rceil^H$ is used when the source granularity can be left implicit (e.g., when we are dealing with a fixed set of granularities having a distinguished bottom granularity).

Another direction of the above transformation is the *down* operation: Let G and H be granularities such that $G \trianglelefteq H$, and z an integer. Define $\lfloor z \rfloor_G^H$ as the set S of labels of granules of G such that $\bigcup_{j \in S} G(j) = H(z)$.[2] This function is useful for finding, e.g., all the days in a month.

2.4 The Periodical Granules Representation

A central issue in temporal reasoning is the possibility of finitely representing infinite granularities. The definition of granularity provided above is general and expressive but it may be impossible to provide a finite representation of some of the granularities. Even labels (i.e., a subset of the integers) do not necessarily have a finite representation.

A solution has been first proposed by Bettini et al. (?). The idea is that most of the commonly used granularities present a periodical behavior; it means that there is a certain pattern that repeats periodically. This feature has been exploited to provide a method for finitely describing granularities. The formal definition is based on the *periodically groups into* relationship.

Definition 5 *A labeled granularity G* groups periodically into *a labeled granularity H ($G \,\bar{\trianglelefteq}\, H$) if $G \trianglelefteq H$ and there exist positive integers N and P such that*

(1) for each label i of H, $i + N$ is a label of H unless $i + N$ is greater than the greatest label of H, and

(2) for each label i of H, if $H(i) = \bigcup_{r=0}^{k} G(j_r)$ and $H(i+N)$ is a non-empty granule of H then $H(i + N) = \bigcup_{r=0}^{k} G(j_r + P)$, and

(3) if $H(s)$ is the first non-empty granule in H (if exists), then $H(s + N)$ is non-empty.

The *groups periodically into* relationship is a special case of the group into characterized by a periodic repetition of the "grouping pattern" of granules of G into granules of H. Its definition may appear complicated but it is actually quite simple. Since G groups into H, any granule $H(i)$ is the union of some granules of G; for instance assume it is the union of the granules $G(a_1), G(a_2), \ldots, G(a_k)$.

[1]Condition (1) of Definition 1.

[2]This definition is different from the one given by Bettini et al (?) since it also considers non contiguous granules of G.

Condition (1) ensures that the label $i + N$ exists (if it not greater than the greatest label of H) while condition (2) ensures that, if $H(i + N)$ is not empty, then it is the union of $G(a_1 + P), G(a_2 + P), \ldots, G(a_k + P)$. We assume that $\forall r = 0 \ldots k$, $(j_r + P) \in \mathcal{L}_G$; if not, the conditions are considered not satisfied. Condition (3) simply says that there is at least one of these repetitions.

We call each pair P and N in Definition 5, a *period length* and its associated *period label distance*. We also indicate with R the number of granules of H corresponding to each groups of P consecutive granules of $\perp$. More formally R is equal to the number of labels of H greater or equal than i and smaller than $i + N$ where i is an arbitrary label of H. Note that R is not affected by the value of i.

The period length and the period label distance are not unique; more precisely, we indicate with P_H^G the period length of H in terms of G and with N_H^G the period label distance of H in terms of G; the form P_H and N_H is used when $G = \perp$. Note that the period length is an integer value. For simplicity we also indicate with one period of a granularity H a set of R consecutive granules of H.

In general, the *periodically groups into* relationship guarantees that granularity H can be finitely described (in terms of granules of G).

Definition 6 *If $G \trianglelefteq H$, then H can be finitely described by providing: (i) a value for P and N; (ii) the set $\mathcal{L}^P$ of labels of H in one period of H; (iii) for each $a \in \mathcal{L}^P$, the finite set S_a of labels of G, such that $H(a) = \bigcup_{i \in S_a} G(i)$; (iv) the labels of first and last non-empty granules in H, if their values are not infinite.*

In this representation, the granules that have labels in $\mathcal{L}^P$ are the only ones that need to be explicitly represented; we call these granules the *explicit granules*.

If a granularity H can be represented as a periodic set of granules of a granularity G, then there exists an infinite number of pairs (P_H^G, N_H^G) for which the periodically groups into relation is satisfied. If the relation is satisfied for a pair (P, N), then it can be proved that it can also be satisfied for each pair $(\alpha P, \alpha N)$ with $\alpha \in \mathbb{N}^+$.

Definition 7 *A periodic representation of a granularity H in terms of G is called* minimal *if the period length P used in the representation has the smallest value among the period lengths appearing in all the pairs (P_H^G, N_H^G) for which H periodically groups into G.*

If H is fully characterized in terms of G, it is possible to derive the composition, in terms of G, of any granule of H. Indeed, if $\mathcal{L}^P$ is the set of labels of H with values in $\{b, \ldots, b + N_H^G - 1\}$, and we assume H to be unbounded, the description of an arbitrary granule $H(j)$ can be obtained by the following formula. Given $j' = [(j - 1) \bmod N_H^G] + 1$ and

$$k = \begin{cases} \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor\right) \cdot N_H^G + j' & \text{if } \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor\right) \cdot N_H^G + j' \geq b \\ \left(\left\lfloor \frac{b-1}{N_H^G} \right\rfloor + 1\right) \cdot N_H^G + j' & \text{otherwise} \end{cases}$$

we have
$$H(j) = \bigcup_{i \in S_k} G\left(P_H^G \cdot \left\lfloor \frac{j-1}{N_H^G} \right\rfloor + i - P_H^G \cdot \left\lfloor \frac{k-1}{N_H^G} \right\rfloor\right).$$

Example 1 *Figure 1 shows granularities* `day` *and* `week_parts` *i.e., the granularity that, for each week, contains a granule for the working days and a granule for the weekend. For the sake of simplicity, we denote* `day` *and* `week_parts` *with* D *and* W *respectively. Since* $D \trianglelefteq W$, W *is fully characterized in terms of* D. *Among different possible representations, in this example we decide to represent* W *in terms of* D *by* $P_W^D = 7$, $N_W^D = 2$, $\mathcal{L}_W^P = \{3, 4\}$, $S_3 = \{8, 9, 10, 11, 12\}$ *and* $S_4 = \{13, 14\}$. *The composition of each granule of* W *can then be easily computed; For example the composition of* $W(6)$ *is given by the formula presented above with* $j' = 2$ *and* $k = 4$. *Hence* $W(6) = D(7{\cdot}2{+}13{-}7{\cdot}1) \cup D(7{\cdot}2{+}14{-}7{\cdot}1) = D(20) \cup D(21)$.

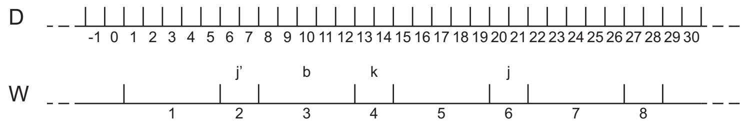


Figure 1: *Periodically groups into* example

3 Calendar Algebra

Several high-level symbolic formalisms have been proposed to represent granularities (?, ?).

In this work we consider the formalism proposed by Ning et al. (?) called *Calendar Algebra*. In this approach a set of algebraic operations is defined; each operation generates a new granularity by manipulating other granularities that have already been generated. The relationships between the operands and the resulting granularities are thus encoded in the operations. All granularities that are generated directly or indirectly from the bottom granularity form a *calendar*, and these granularities are related to each other through the operations that define them. In practice, the choices for the bottom granularity include `day`, `hour`, `second`, `microsecond` and other granularities, depending on the accuracy required in each application context.

In the following we illustrate the calendar algebra operations presented by Ning et al. (?) together with some restrictions introduced by Bettini et al. (?).

3.1 The Grouping-Oriented Operations

The calendar algebra consists of the following two kinds of operations: the *grouping-oriented operations* and the *granule-oriented operations*. The grouping-oriented operations group certain granules of a granularity together to form new granules in a new granularity.

3.1.1 The Grouping Operation

Let G be a full-integer labeled granularity, and m a positive integer. The grouping operation $\mathit{Group}_m(G)$ generates a new granularity G' by partitioning the granules of G into m-granule groups and making each group a granule of the resulting granularity. More precisely, $G' = \mathit{Group}_m(G)$ is the granularity such that for each integer i,

$$G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i\cdot m} G(j).$$

For example, given granularity `day`, granularity `week` can be generated by the calendar algebra expression $\texttt{week} = \mathit{Group}_7(\texttt{day})$ if we assume that $\texttt{day}(1)$ corresponds to Monday, i.e., the first day of a week.

3.1.2 The Altering-tick Operation

Let G_1, G_2 be full-integer labeled granularities, and l, k, m integers, where G_2 partitions G_1, and $1 \leq l \leq m$. The altering-tick operation $\mathit{Alter}^m_{l,k}(G_2, G_1)$ generates a new granularity by periodically expanding or shrinking granules of G_1 in terms of granules of G_2. Since G_2 partitions G_1, each granule of G_1 consists of some contiguous granules of G_2. The granules of G_1 can be partitioned into m-granule groups such that $G_1(1)$ to $G_1(m)$ are in one group, $G_1(m+1)$ to $G_1(2m)$ are in the following group, and so on. The goal of the altering-tick operation is to modify the granules of G_1 so that the l-th granule of every m-granule group will have $|k|$ additional (or fewer when $k < 0$) granules of G_2. For example, if G_1 represents 30-day groups (i.e., $G_1 = \mathit{Group}_{30}(\texttt{day})$) and we want to add a day to every 3-rd month (i.e., to make March to have 31 days), we may perform $\mathit{Alter}^{12}_{3,1}(\texttt{day}, G_1)$.

The altering-tick operation can be formally described as follows. For each integer i such that $G_1(i) \neq \emptyset$, let b_i and t_i be the integers such that $G_1(i) = \cup_{j=b_i}^{t_i} G_2(j)$ (the integers b_i and t_i exist because G_2 partitions G_1). Then $G' = \mathit{Alter}^m_{l,k}(G_2, G_1)$ is the granularity such that for each integer i, let $G'(i) = \emptyset$ if $G_1(i) = \emptyset$, and otherwise let

$$G'(i) = \bigcup_{j=b'_i}^{t'_i} G_2(j),$$

where

$$b'_i = \begin{cases} b_i + (h-1)\cdot k, & \text{if } i = (h-1)\cdot m + l, \\ b_i + h \cdot k, & \text{otherwise,} \end{cases}$$

$$t'_i = t_i + h \cdot k,$$

and

$$h = \left\lfloor \frac{i-l}{m} \right\rfloor + 1.$$

Example 2 *Figure 2 shows an example of the* `Alter` *operation. Granularity G_1 is defined by $G_1 = Group_5(G_2)$ and granularity G' is defined by $G' = Alter^2_{2,-1}(G_2, G_1)$, which means shrinking the second one of every two granules of G_1 by one granule of G_2.*

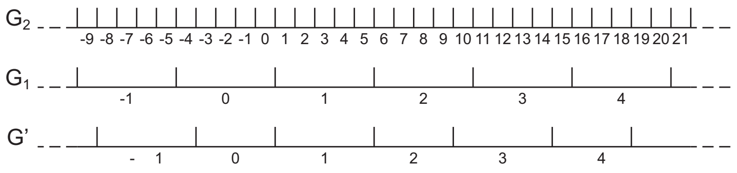


Figure 2: *Altering-tick* operation example

The original definition of altering-tick given by Ning et al. (?) as reported above, has the following problems when an arbitrary negative value for k is used: (1) It allows the definition of a G' that is not a full-integer labeled granularity and (2) It allows the definition of a G' that does not even satisfy the definition of granularity. In order to avoid this undesired behavior, we impose the following restriction:

$$k > -(mindist(G1, 2, G2) - 1)$$

where $mindist()$ is formally defined by Bettini et al. (?).

Intuitively, $mindist(G1, 2, G2)$ represents the minimum distance (in terms of granules of $G2$) between two consecutive granules of $G1$.

3.1.3 The Shift Operation

Let G be a full-integer labeled granularity, and m an integer. The shifting operation $Shift_m(G)$ generates a new granularity G' by shifting the labels of G by m positions. More formally, $G' = Shift_m(G)$ is the granularity such that for each integer i, $G'(i) = G(i - m)$. Note that G' is also full-integer labeled.

3.1.4 The Combining Operation

Let G_1 and G_2 be granularities with label sets $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$ respectively. The combining operation $Combine(G_1, G_2)$ generates a new granularity G' by combining all the granules of G_2 that are included in one granule of G_1 into one granule of G'. More formally, for each $i \in \mathcal{L}_1$, let $s(i) = \emptyset$ if $G_1(i) = \emptyset$, and otherwise let $s(i) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$. Then $G' = Combine(G_1, G_2)$ is the granularity with the label set $\mathcal{L}_{G'} = \{i \in \mathcal{L}_{G_1} | s(i) \neq \emptyset\}$ such that for each i in $\mathcal{L}_{G'}$, $G'(i) = \bigcup_{j \in s(i)} G_2(j)$.

As an example, given granularities `b-day` and `month`, the granularity for business months can be generated by `b-month` $=$ $Combine($`month`$,$ `b-day`$)$.

3.1.5 The Anchored Grouping Operation

Let G_1 and G_2 be granularities with label sets $\mathcal{L}_{G_1}$ and $\mathcal{L}_{G_2}$ respectively, where G_2 is a label-aligned subgranularity of G_1, and G_1 is a full-integer labeled granularity. The anchored grouping operation $\textit{Anchored-group}(G_1, G_2)$ generates a new granularity G' by combining all the granules of G_1 that are between two granules of G_2 into one granule of G'. More formally, $G' = \textit{Anchored-group}(G_1, G_2)$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_2}$ such that for each $i \in \mathcal{L}_{G'}$, $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ where i' is the next label of G_2 after i.

For example, each academic year at a certain university begins on the last Monday in August, and ends on the day before the beginning of the next academic year. Then, the granularity corresponding to the academic years can be generated by $AcademicYear = \textit{Anchored-group}(\texttt{day}, \texttt{lastMondayOfAugust})$.

3.2 The Granule-Oriented Operations

Differently from the grouping-oriented operations, the granule-oriented operations do not modify the granules of a granularity, but rather enable the selection of the granules that should remain in the new granularity.

3.2.1 The Subset Operation

Let G be a granularity with label set $\mathcal{L}_G$, and m, n integers such that $m \leq n$. The subset operation $G' = \textit{Subset}_m^n(G)$ generates a new granularity G' by taking all the granules of G whose labels are between m and n. More formally, $G' = \textit{Subset}_m^n(G)$ is the granularity with the label set $\mathcal{L}_{G'} = \{i \in \mathcal{L}_G \mid m \leq i \leq n\}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G(i)$. For example, given granularity `year`, all the years in the 20th century can be generated by $\texttt{20CenturyYear} = \textit{Subset}_{1900}^{1999}(\texttt{year})$. Note that G' is a label-aligned subgranularity of G, and G' is not a full-integer labeled granularity even if G is. We also allow the extensions of setting $m = -\infty$ or $n = \infty$ with semantics properly extended.

3.2.2 The Selecting Operations

The selecting operations are all binary operations. They generate new granularities by selecting granules from the first operand in terms of their relationship with the granules of the second operand. The result is always a label-aligned subgranularity of the first operand granularity.

There are three selecting operations: *select-down*, *select-up* and *select-by-intersect*. To facilitate the description of these operations, the $\Delta_k^l(S)$ notation is used. Intuitively, if S is a set of integers, $\Delta_k^l(S)$ selects l elements starting from the k-th one (for a formal description of the Δ operator see (?)).

Select-down operation. For each granule $G_2(i)$, there exits a set of granules of G_1 that is contained in $G_2(i)$. The operation $\textit{Select-down}_k^l(G_1, G_2)$, where $k \neq 0$ and $l > 0$ are integers, selects granules of G_1 by using $\Delta_k^l(\cdot)$ on each set of granules (actually their labels) of G_1 that are contained in one granule of G_2. More formally, $G' = \textit{Select-down}_k^l(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \cup_{i \in \mathcal{L}_{G_2}} \Delta_k^l(\{j \in \mathcal{L}_{G_1} \mid \emptyset \neq G_1(j) \subseteq G_2(i)\}),$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, Thanksgiving days are the fourth Thursdays of all Novembers; if `Thursday` and `November` are given, it can be generated by $\texttt{Thanksgiving} = \textit{Select-down}_4^1(\texttt{Thursday}, \texttt{November})$.

Select-up operation. The select-up operation $\textit{Select-up}(G_1, G_2)$ generates a new granularity G' by selecting the granules of G_1 that contain one or more granules of G_2. More formally, $G' = \textit{Select-up}(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \{i \in \mathcal{L}_{G_1} | \exists j \in \mathcal{L}_{G_2} (\emptyset \neq G_2(j) \subseteq G_1(i)), \}$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, given granularities `Thanksgiving` and `week`, the weeks that contain Thanksgiving days can be defined by $\texttt{ThanxWeek} = \textit{Select-up}(\texttt{week}, \texttt{Thanksgiving})$.

Select-by-intersect operation. For each granule $G_2(i)$, there may exist a set of granules of G_1, each intersecting $G_2(i)$. The $\textit{Select-by-intersect}_k^l(G_1, G_2)$ operation, where $k \neq 0$ and $l > 0$ are integers, selects granules of G_1 by applying $\Delta_k^l(\cdot)$ operator to all such sets, generating a new granularity G'. More formally, $G' = \textit{Select-by-intersect}_k^l(G_1, G_2)$ is the granularity with the label set

$$\mathcal{L}_{G'} = \cup_{i \in \mathcal{L}_{G_2}} \Delta_k^l(\{j \in \mathcal{L}_{G_1} \mid G_1(j) \cap G_2(i) \neq \emptyset\}),$$

and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$. For example, given granularities `week` and `month`, the granularity consisting of the first week of each month (among all the weeks intersecting the month) can be generated by $\texttt{FirstWeekOfMonth} = \textit{Select-by-intersect}_1^1(\texttt{week}, \texttt{month})$.

3.2.3 The Set Operations

In order to have the set operations as a part of the calendar algebra and to make certain computations easier, we restrict the operand granularities participating in the set operations so that the result of the operation is always a valid granularity: the set operations can be defined on G_1 and G_2 only if there exists a granularity H such that G_1 and G_2 are both label-aligned subgranularities of H. In the following, we describe the union, intersection, and difference operations of G_1 and G_2, assuming that they satisfy the requirement.

Union. The union operation $G_1 \cup G_2$ generates a new granularity G' by collecting all the granules from both G_1 and G_2. More formally, $G' = G_1 \cup G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cup \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$,

$$G'(i) = \begin{cases} G_1(i), & i \in \mathcal{L}_1, \\ G_2(i), & i \in \mathcal{L}_2 - \mathcal{L}_1. \end{cases}$$

For example, given granularities `Sunday` and `Saturday`, the granularity of the weekend days can be generated by `WeekendDay` $=$ `Sunday` $\cup$ `Saturday`.

Intersection. The intersection operation $G_1 \cap G_2$ generates a new granularity G' by taking the common granules from both G_1 and G_2. More formally, $G' = G_1 \cap G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cap \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$ (or equivalently $G_2(i)$).

Difference. The difference operation $G_1 \setminus G_2$ generates a new granularity G' by excluding the granules of G_2 from those of G_1. More formally, $G' = G_1 \setminus G_2$ is the granularity with the label set $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \setminus \mathcal{L}_{G_2}$, and for each $i \in \mathcal{L}_{G'}$, $G'(i) = G_1(i)$.

4 From Calendar Algebra to Periodical Set

In this section we first describe the overall conversion process and then we report the formulas specific for the conversion of each calendar algebra operation. Finally, we present a procedure for relabeling the resulting granularity, a sketch complexity analysis and some considerations about the period length minimality.

4.1 The Conversion Process

Our final goal is to provide a correct and effective way to convert calendar expressions into periodical representations. Under appropriate limitations, for each calendar algebra operation, if the periodical descriptions of the operand granularities are known, it is possible to compute the periodical characterization of the resulting granularity.

This result allows us to calculate, for any calendar, the periodical description of each granularity in terms of the bottom granularity. In fact, by definition, the bottom granularity is fully characterized; hence it is possible to compute the periodical representation of all the granularities that are obtained from operations applied to the bottom granularity. Recursively, the periodical description of all the granularities can be obtained.

The calendar algebra presented in the previous section can represent all the granularities that are periodical with finite exceptions (i.e., any granularity G such that bottom groups periodically with finite exceptions into G). Since with the periodical representations defined in Section 2 it is not possible to express the finite exceptions, we need to restrict the calendar algebra so that it cannot represent them. This implies allowing the *Subset* operation to be only used as the last step of deriving a granularity. Note that in the calendar algebra presented by Ning et al. (?) there was an extension to the altering-tick operation to allow the usage of ∞ as the m parameter (i.e., $G' = Alter^{\infty}_{l,k}(G_2, G_1)$); the resulting granularity has a single exception hence is not periodic. This extension is disallowed here in order to generate periodical granularities only (without finite exceptions).

The conversion process can be divided into three steps: in the first one the period length and period label distance are computed; in the second we derive the set $\mathcal{L}^P$ of labels in one period, and in the last one the composition of the explicit granules is computed. For each operation we identify the correct formulas and algorithms for the three steps.

The **first step** consists in computing the period length and the period label distance of the resulting granularity. Those values are calculated as a function of the parameters (e.g. the "grouping factor" m, in the *Group* operation) and the operand granularities (actually their period lengths and period label distances).

The **second step** in the conversion process is the identification of the label set of the resulting granularity. In Section 2.4 we pointed out that in order to fully characterize a granularity it is sufficient to identify the labels in any period of the granularity. In spite of this theoretical result, to perform the computations required by each operation we need the explicit granules of the operand granularities to be "aligned". There are two possible approaches: the first one consist in computing the explicit granules in any period and then recalculate the needed granules in the correct position in order to eventually align them. The second one consists in aligning all the periods containing the explicit granules with a fixed granule in the bottom granularity. After considering both possibilities, for performance reasons, we decided to adopt the second approach. We decided to use $\perp(1)$ as the "alignment point" for all the granularities. A formal definition of the used formalism follows.

Let G be a granularity and i be the smallest positive integer such that $\lceil i \rceil^G$ is defined. We call $l_G = \lceil i \rceil^G$ and $\overline{\mathcal{L}}_G$ the set of labels of G contained in $l_G \dots l_G + N_G - 1$. Note that this definition of $\overline{\mathcal{L}}_G$ is an instance of the definition of $\mathcal{L}^P$ given in Section 2.4. The definition of $\overline{\mathcal{L}}_G$ provided here is useful for representing G and actually the final goal of this step is to compute $\overline{\mathcal{L}}_G$; however $\overline{\mathcal{L}}_G$ is not suitable for performing the computations. The problem is that if $G(l_G)$ starts before $\perp(1)$ (i.e., $min(\lfloor l_G \rfloor^G) < 1$) then the granule $G(l_G + N_G)$ begins at P_G or before P_G, and hence $G(l_G + N_G)$ is necessary for the computations; however $l_G + N_G \notin \overline{\mathcal{L}}_G$.

To solve the problem we introduce the symbol $\hat{\mathcal{L}}_G$ to represent the set of all labels of granules of G that cover one in $\perp(1) \dots \perp(P_G)$. It is easily seen that if $G(l_G)$ does not cover $\perp(0)$, then $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G$, otherwise $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G \cup \{l_G + N_G\}$. Therefore the conversion between $\overline{\mathcal{L}}$ and $\hat{\mathcal{L}}$ and vice versa is immediate.

The notion of $\hat{\mathcal{L}}$ is still not enough to perform the computations. The problem is that when a granularity G is used as an operand in an operation, the period length of the resulting granularity G' is generally bigger than the period length of G. Therefore it is necessary to extend the notion of $\hat{\mathcal{L}}_G$ to the period length $P_{G'}$ of G' using $P_{G'}$ in spite of P_G in the definition of $\hat{\mathcal{L}}$. The symbol used for this notion is $\hat{\mathcal{L}}_G^{P_{G'}}$.

The idea is that when G is used as the operand in an operation that generates G', $\hat{\mathcal{L}}_G^{P_{G'}}$ is computed from $\overline{\mathcal{L}}_G$. This set is then used by the formula that we provide below to compute $\overline{\mathcal{L}}_{G'}$.

The computation of $\overline{\mathcal{L}}_{G'}$ is performed as follows: if G' is defined by an operation that returns a full-integer labeled granularity, then it is sufficient to compute the value of l'_G. Indeed it is easily seen that $\overline{\mathcal{L}}_{G'} = \{i \in \mathbb{Z} | l'_G \leq i \leq l'_G + N_{G'} - 1\}$. If G' is defined by any other algebraic operation, we provide the formulas to compute $\hat{\mathcal{L}}_{G'}$; from $\hat{\mathcal{L}}_{G'}$ we easily derive $\overline{\mathcal{L}}_{G'}$.

Example 3 *Figure 3 shows granularities $\perp$, G and H; it is clear that $P_G = P_H = 4$ and $N_G = N_H = 3$. Moreover, $l_G = l_H = 6$ and therefore $\overline{\mathcal{L}}_G = \overline{\mathcal{L}}_H = \{6, 7\}$. Since $0 \notin \lfloor 6 \rfloor^G$ then $\hat{\mathcal{L}}_G = \overline{\mathcal{L}}_G$. On the other hand, since $0 \in \lfloor 6 \rfloor^H$, then $\hat{\mathcal{L}}_H = \overline{\mathcal{L}}_H \cup \{6+3\}$.*

Suppose that a granularity G' has period length $P_{G'} = 8$; then $\hat{\mathcal{L}}_G^{P'_G} = \{6, 7, 9, 10\}$ and $\hat{\mathcal{L}}_H^{P_{G'}} = \{6, 7, 9, 10, 12\}$.

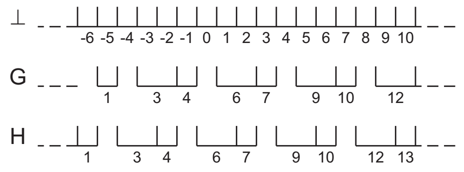


Figure 3: $\overline{\mathcal{L}}$, l, $\hat{\mathcal{L}}$ and $\hat{\mathcal{L}}^{P_{G'}}$ examples

The **third** (and last) **step** of the conversion process is the computation of the composition of the explicit granules. Once $\overline{\mathcal{L}}_{G'}$ has been computed, it is sufficient to apply, for each label of $\overline{\mathcal{L}}_{G'}$ the formulas presented in Chapter 3.

In Sections 4.3 to 4.10 we show, for each calendar algebra operation, how to compute the first and second conversion steps.

4.2 Computability Issues

In some of the formulas presented below it is necessary to compute the set S of labels of a granularity G such that $\forall i \in S\ G(i) \subseteq H(j)$ where H is a granularity and j is a specific label of H. Since $\mathcal{L}_G$ contains an infinite number of labels, it is not possible to check, $\forall i \in \mathcal{L}_G$ if $G(i) \subseteq H(j)$. However it is easily seen that $\forall i \in S\ \exists k$ s.t. $G(\lceil k \rceil^G) \subseteq H(j)$. Therefore $\forall i \in S\ \exists k$ s.t. $G(\lceil k \rceil^G)$ is defined and $k \in \lfloor j \rfloor^H$.

Therefore we compute the set S by considering all the labels i of $\mathcal{L}_G$ s.t. $\exists n \in \lfloor j \rfloor^H$ s.t. $\lceil n \rceil^G = i$ and $G(i) \subseteq H(j)$. Since the set $\lfloor j \rfloor^H$ is finite[3], the computation can be performed in a finite time. The consideration is analogous if S is the set such that $\forall i \in S\ G(i) \supseteq H(j)$ or $\forall i \in S\ (G(i) \cap H(j) \neq \emptyset)$.

4.3 The Group Operation

Proposition 1 *If $G' = Group_m(G)$, then:*

[3]With the calendar algebra it is not possible to define granularities having granules that maps to an infinite set of time instants.

1. $P_{G'} = \frac{P_G \cdot m}{GCD(m, N_G)}$ and $N_{G'} = \frac{N_G}{GCD(m, N_G)}$;

2. $l_{G'} = \left(\left\lfloor \frac{l_G - 1}{m} \right\rfloor + 1\right)$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i \cdot m} G(j)$.

Example 4 *Figure 4 shows an example of the group operation: $G' = Group_3(G)$. Since $P_G = 1$ and $N_G = 1$, then $P_{G'} = 3$ and $N_G = 1$. Moreover, since $\overline{\mathcal{L}}_G = \{-7\}$, then $l_G = -7$ and therefore $l_{G'} = -2$ and $\overline{\mathcal{L}}_{G'} = \{-2\}$. Finally $G'(-2) = G(-8) \cup G(-7) \cup G(-6)$ i.e., $G'(-2) = \perp(0) \cup \perp(1) \cup \perp(2)$.*

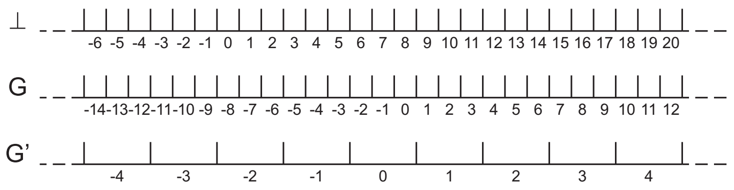


Figure 4: *Group* operation example

4.4 The Altering-tick Operation

Proposition 2 *If $G' = Alter^m_{l,k}(G_2, G_1)$ then:*

1.

$$N_{G'} = lcm\left(N_{G_1}, m, \frac{P_{G_2} \cdot N_{G_1}}{GCD(P_{G_2} \cdot N_{G_1}, P_{G_1})}, \frac{N_{G_2} \cdot m}{GCD(N_{G_2} \cdot m, |k|)}\right)$$

and

$$P_{G'} = \left(\frac{N_{G'} \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{N_{G'} \cdot k}{m}\right) \cdot \frac{P_{G_2}}{N_{G_2}}$$

2. $l_{G'} = \lceil l_{G_2} \rceil^{G'}_{G_2}$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j=b'_i}^{t'_i} G(j)$ *where b'_i and t'_i are defined in Section 3.1.2.*

Referring to step 2., note that when computing $l_{G'}$ the explicit characterization of the granules of G' is still unknown. To perform the operation $\lceil l_{G_2} \rceil^{G'}_{G_2}$ we need to know at least the explicit granules of one of its periods. We choose to compute the granules labeled by $1 \ldots N_{G'}$. When $l_{G'}$ is derived, the granules labeled by $l_{G'} \ldots l_{G'} + N_{G'} - 1$ will be computed so that the explicit granules are aligned to $\perp(1)$ as required.

Example 5 *Figure 5 shows an example of the altering-tick operation: $G' = Alter^3_{2,1}(G_2, G_1)$. Since $P_{G_1} = 4$, $N_{G_1} = 1$, $P_{G_2} = 4$ and $N_{G_2} = 2$, then $N_{G'} = 6$ and $P_{G'} = 28$. Moreover, since $\overline{\mathcal{L}}_{G_2} = \{-10, -9\}$, then $l_{G_2} = -10$*

and therefore $l_{G'} = \lceil -10 \rceil_{G_2}^{G'} = -4$ *and hence* $\overline{\mathcal{L}}_{G_2} = \{-4, -3, \ldots, 0, 1\}$. *Finally* $G'(-4) = G_1(-11) \cup G_1(-10) \cup G_1(-9) = \perp(-1) \cup \perp(0) \cup \perp(1) \cup \perp(3) \cup \perp(4)$; *analogously we derive* $G'(-3)$, $G'(-2)$, $G'(-1)$, $G'(0)$ *and* $G'(1)$.

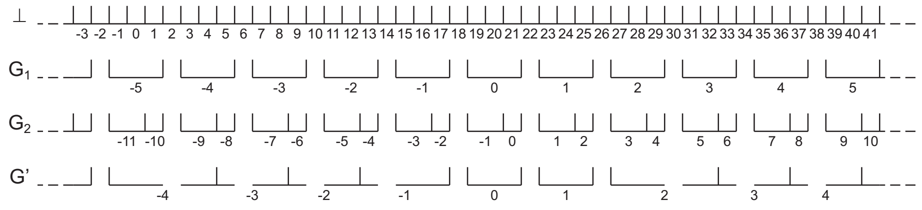


Figure 5: *Alter* operation example

4.5 The Shift Operation

Proposition 3 *If* $G' = Shift_m(G)$, *then:*

1. $P_{G'} = P_{G_1}$ *and* $N_{G'} = N_{G_1}$;
2. $l_{G'} = l_G + m$;
3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = G(i - m)$.

Example 6 *The shifting operation can easily model time differences. Suppose granularity* `USEast-Hour` *stands for the hours of US Eastern Time. Since the hours of the US Pacific Time are 3 hours later than those of US Eastern Time, the hours of US Pacific Time can be generated by* `USPacific-Hour`$=$ $Shift_{-3}$(`USEast-Hour`).

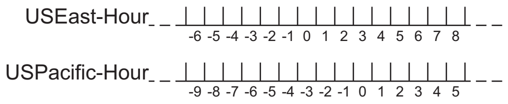


Figure 6: *Shift* operation example

4.6 The Combining Operation

Proposition 4 *Given* $G' = Combining(G_1, G_2)$, *then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$;
2. $\forall i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ *let be* $\widetilde{s}(i) = \{j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$; *then* $\hat{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \widetilde{s}(i) \neq \emptyset\}$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j \in s(i)} G_2(j)$.

Example 7 *Figure 7 shows an example of the combining operation:* $G' = Combine(G_1, G_2)$. *Since* $P_{G_1} = 6$, $N_{G_1} = 2$, $P_{G_2} = 4$ *and* $N_{G_2} = 2$, *then* $P_{G'} = 12$ *and* $N_{G'} = 4$. *Moreover, since* $\overline{\mathcal{L}}_{G_1} = \{1\}$ *and* $0 \in \lfloor 1 \rfloor^{G_1}$, *then* $\hat{\mathcal{L}}_{G_1} = \{1, 3\}$ *and hence* $\hat{\mathcal{L}}_{G_1}^{P_{G'}} = \{1, 3, 5\}$. *Since* $\tilde{s}(i) \neq \emptyset$ *for* $i \in \{1, 3, 5\}$, *then* $\hat{\mathcal{L}}_{G'} = \{1, 3, 5\}$*; moreover, since* $0 \in \lfloor 1 \rfloor^{G'}$, *then* $\overline{\mathcal{L}}_{G'} = \{1, 3\}$. *Finally* $s(1) = \{-1, 0\}$ *and* $s(3) = \{2, 3\}$*; consequently,* $G'(1) = G_2(-1) \cup G_2(0)$ *i.e.,* $G'(1) = \perp(-1) \cup \perp(0) \cup \perp(1)$ *and* $G'(3) = G_2(2) \cup G_2(3)$ *i.e.,* $G'(3) = \perp(4) \cup \perp(5) \cup \perp(7)$.

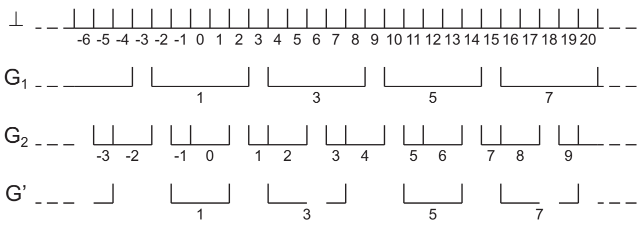


Figure 7: *Combine* operation example

4.7 The Anchored Grouping Operation

Proposition 5 *Given* $G' = Anchored\text{-}group(G_1, G_2)$, *then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_2}}{P_{G_2}}$;

2.

$$\hat{\mathcal{L}}_{G'} = \begin{cases} \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{if } l_{G_2} = l_{G_1}, \\ \{l'_{G_2}\} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{otherwise,} \end{cases}$$

where l'_{G_2} *is the greatest among the labels of* $\mathcal{L}_{G_2}$ *that are smaller than* l_{G_2}.

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ *where* i' *is the next label of* G_2 *after* i.

Example 8 *Figure 8 shows an example of the anchored grouping operation: the* `USweek` *(i.e., a week starting with a* `Sunday`*) is defined by the operation* Anchored-group*(*`day`*,* `Sunday`*). Since* $P_{\texttt{day}} = 1$ *and* $P_{\texttt{Sunday}} = 7$, *then the period length of* `USweek` *is 7. Moreover since* $l_{\texttt{day}} = 11$, $l_{\texttt{Sunday}} = 14$ *and* $\hat{\mathcal{L}}_{\texttt{Sunday}}^{P_{\texttt{USweek}}} = \{14\}$, *then* $\hat{\mathcal{L}}_{\texttt{USweek}} = \{7\} \cup \{14\}$. *Clearly, since* $0 \in \lfloor 7 \rfloor^{\texttt{USweek}}$ *then* $\overline{\mathcal{L}}_{\texttt{USweek}} = \{7\}$. *Finally,* $\texttt{USweek}(7) = \bigcup_{j=7}^{13} \texttt{day}(j) = \bigcup_{k=-3}^{3} \perp(k)$.

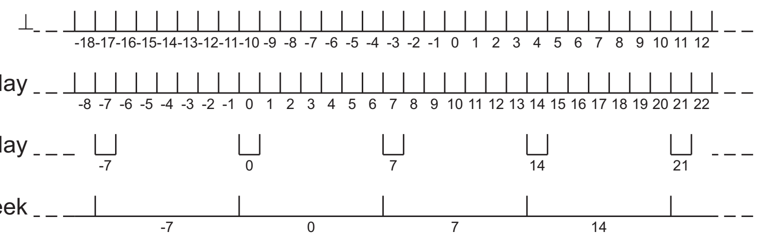


Figure 8: *Anchored Grouping* operation example

4.8 The Subset Operation

The *Subset* operation only modifies the operand granularity by introducing the bounds. The period length, the period label distance, $\overline{\mathcal{L}}$ and the composition of the explicit granules are not affected.

4.9 The Selecting Operations

4.9.1 The Select-down Operation

Proposition 6 *Given $G' = \textit{Select-down}^l_k(G_1, G_2)$, then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1}, P_{G_2}) \cdot N_{G_1}}{P_{G_1}}$;

2. $\forall i \in \mathcal{L}_{G_2}$ *let*

$$A(i) = \Delta^l_k \left(\{ j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i) \} \right).$$

Then

$$\hat{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}^{P_{G'}}_{G_2}} \left\{ a \in A(i) | a \in \hat{\mathcal{L}}^{P_{G'}}_{G_1} \right\};$$

3. $\forall i \in \overline{\mathcal{L}}_{G'}$ $G'(i) = G_1(i)$.

Example 9 *Figure 9 shows an example of the Select-down operation in which granularity G' is defined as: $G' = \textit{Select-down}^1_2(G_1, G_2)$. Since $P_{G_1} = 4$, $N_{G_1} = 2$ and $P_{G_2} = 6$ then $P_{G'} = 12$ and $N_{G'} = 6$. Moreover, since $\overline{\mathcal{L}}_{G_2} = \{-3\}$ and $0 \in \lfloor -3 \rfloor^{G_2}$, then $\hat{\mathcal{L}}_{G_2} = \{-3, -2\}$ and $\hat{\mathcal{L}}^{P_{G'}}_{G_2} = \{-3, -2, -1\}$. Intuitively, $A(-3) = \{-5\}$, $A(-2) = \{-2\}$ and $A(-1) = \{1\}$. Hence $\hat{\mathcal{L}}_{G'} = \{-5, -2, 1\}$ and therefore, since $0 \in \lfloor -5 \rfloor^{G'}$, $\overline{\mathcal{L}}_{G'} = \{-5, -2\}$. Finally $G'(-5) = G_1(-5) = \perp(0) \cup \perp(1)$ and $G'(-2) = G_1(-2) = \perp(6)$.*

4.9.2 The Select-up Operation

Proposition 7 *Given $G' = \textit{Select-up}(G_1, G_2)$, then:*

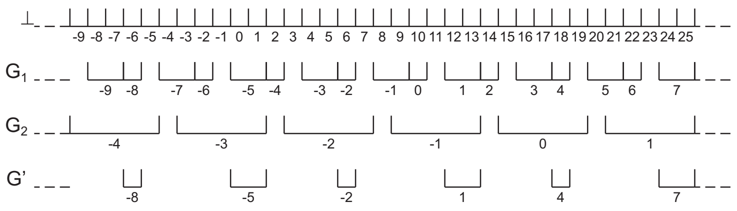


Figure 9: *Select-down* operation example

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1},P_{G_2})\cdot N_{G_1}}{P_{G_1}}$;

2.

$$\hat{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \exists j \in \mathcal{L}_{G_2}\ s.t. \emptyset \neq G_2(j) \subseteq G_1(i)\};$$

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = G_1(i)$.

Example 10 *Figure 10 shows an example of the Select-up operation:* $G' =$ *Select-up*(G_1, G_2). *Since* $P_{G_1} = 6$, $N_{G_1} = 3$ *and* $P_{G_2} = 4$ *then* $P_{G'} = 12$ *and* $N_{G'} = 6$. *Moreover, since* $\overline{\mathcal{L}}_{G_1} = \{-3, -2, -1\}$ *and* $0 \in \lfloor -3 \rfloor^{G_2}$, *then* $\hat{\mathcal{L}}_{G_1} = \{-3, -2, -1, 0\}$ *and* $\hat{\mathcal{L}}_{G_1}^{P'_G} = \{-3, -2, -1, 0, 1, 2, 3\}$. *Since* $G_1(-3) \supseteq G_2(-6)$, $G_1(-1) \supseteq G_2(-4)$ *and* $G_1(3) \supseteq G_2(0)$ *then* $\hat{\mathcal{L}}_{G'} = \{-3, -1, 3\}$ *and, since* $0 \in \lfloor -3 \rfloor^{G'}$, *then* $\overline{\mathcal{L}}_{G'} = \{-3, 1\}$ *Finally* $G'(-3) = G_1(-3) = \perp(0) \cup \perp(1)$ *and* $G'(-1) = G_1(-1) = \perp(4)$.

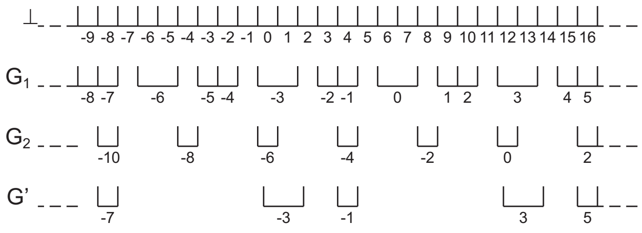


Figure 10: *Select-up* operation example

4.9.3 The Select-by-intersect Operation

Proposition 8 *Given* $G' =$ *Select-by-intersect*$_k^l(G_1, G_2)$, *then:*

1. $P_{G'} = lcm(P_{G_1}, P_{G_2})$ *and* $N_{G'} = \frac{lcm(P_{G_1},P_{G_2})N_{G_1}}{P_{G_1}}$;

2. *then* $\forall i \in \mathcal{L}_{G_2}$ *let*

$$A(i) = \Delta_k^l\left(\{j \in \mathcal{L}_{G_1} | G_1(j) \cap G_2(i) \neq \emptyset\}\right).$$

then

$$\hat{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}} \left\{ a \in A(i) | a \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} \right\}.$$

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = G_1(i)$.

Example 11 *Figure 11 shows an example of the Select-by-intersect operation in which $G' =$ Select-by-intersect$_2^1(G_1, G_2)$. Since $P_{G_1} = 4$, $N_{G_1} = 2$ and $P_{G_2} = 6$ then $P_{G'} = 12$ and $N_{G'} = 6$. Moreover, since $\overline{\mathcal{L}}_{G_2} = \{-3\}$ and $0 \in \lfloor -3 \rfloor^{G_2}$, then $\hat{\mathcal{L}}_{G_2} = \{-3, -2\}$ and $\hat{\mathcal{L}}_{G_2}^{P_{G'}} = \{-3, -2, -1\}$. Intuitively, $A(-3) = \{-6\}$, $A(-2) = \{-2\}$ and $A(-1) = \{0\}$. Hence $\hat{\mathcal{L}}_{G'} = \{-2, 0\}$ and therefore, since $0 \notin \lfloor -5 \rfloor^{G'}$, then $\overline{\mathcal{L}}_{G'} = \{-2, 0\}$. Finally $G'(-2) = G_1(-2) = \perp(6)$ and $G'(0) = G_1(0) = \perp(10)$.*

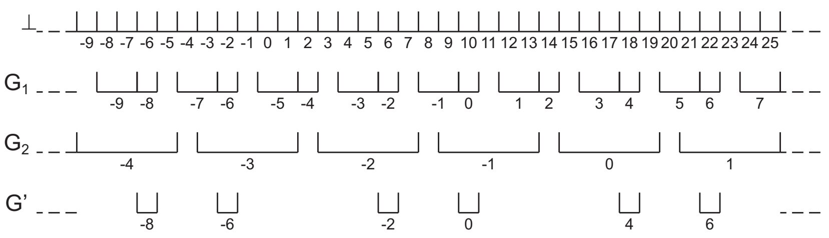


Figure 11: *Select-by-intersect* operation example

4.10 The Set Operations

Since a set operation is valid if the granularities used as argument are both labeled aligned granularity of another granularity, the following property is used.

Proposition 9 *If G is a labeled aligned subgranularity of H, then $\frac{N_G}{P_G} = \frac{N_H}{P_H}$.*

Proposition 10 *Given $G' = G_1 \cup G_2$, $G'' = G_1 \cap G_2$ and $G''' = G_1 \setminus G_2$, then:*

1. $P_{G'} = P_{G''} = P_{G'''} = lcm(P_{G_1}, P_{G_2})$ *and*
   $N_{G'} = N_{G''} = N_{G'''} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}} = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$;

2. $\hat{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G_1}^{P_{G'}} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}$; $\hat{\mathcal{L}}_{G''} = \hat{\mathcal{L}}_{G_1}^{P_{G''}} \cap \hat{\mathcal{L}}_{G_2}^{P_{G''}}$; $\hat{\mathcal{L}}_{G'''} = \hat{\mathcal{L}}_{G_1}^{P_{G'''}} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'''}}$;

3. $\forall i \in \overline{\mathcal{L}}_{G'}\ G'(i) = \begin{cases} G_1(i), & i \in \mathcal{L}_{G_1} \\ G_2(i), & \textit{otherwise}, \end{cases}$

   $\forall i \in \overline{\mathcal{L}}_{G''}\ G''(i) = G_1(i)$ *and* $\forall i \in \overline{\mathcal{L}}_{G'''}\ G'''(i) = G_1(i)$

Example 12 *Figure 12 shows an example of the set operations. Note that both G_1 and G_2 are labeled aligned subgranularities of H. Then $G' = G_1 \cup G_2$, $G'' = G_1 \cap G_2$ and $G''' = G_1 \setminus G_2$. Since $P_{G_1} = P_{G_2} = 6$ and $N_{G_1} = N_{G_2} = 6$ then $P_{G'} = P_{G''} = P_{G'''} = 6$ and $N_{G'} = N_{G''} = N_{G'''} = 2$. Moreover, since $\hat{\mathcal{L}}_{G_1} = \{1,2\}$ and $\hat{\mathcal{L}}_{G_2} = \{2,3\}$, then $\hat{\mathcal{L}}_{G'} = \{1,2,3\}$, $\hat{\mathcal{L}}_{G''} = \{2\}$ and $\hat{\mathcal{L}}_{G'''} = \{1\}$. Finally $G'(1) = G_1(1)$, $G'(2) = G_1(2)$ and $G'(3) = G_2(3)$; $G''(2) = G_1(2)$ and $G'''(1) = G_1(1)$.*

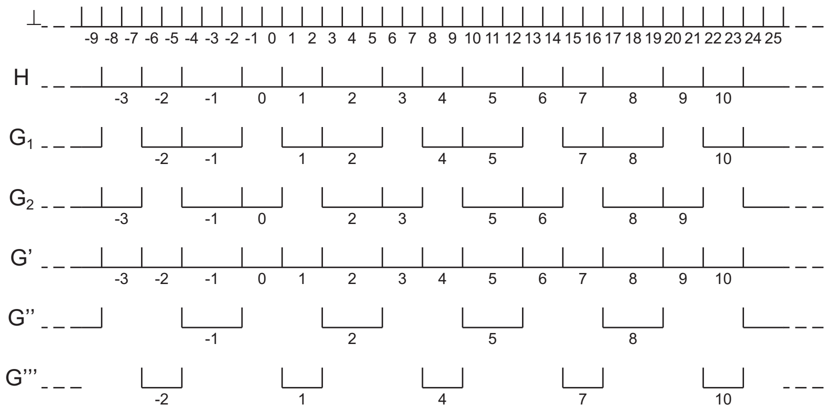


Figure 12: *Set* operations example

4.11 Relabeling

Granularity processing algorithms are much simpler if restricted to operate on full-integer labeled granularities. Moreover, a further simplification is obtained by using only the positive integers as the set of labels (i.e., $\mathcal{L} = \mathbb{Z}^+$).

In this section we show how to relabel a granularity G to obtain a full-integer labeled granularity G'. A granularity G'' such that $\mathcal{L}_{G''} = \mathbb{Z}^+$ can be obtained by using $G'' = Subset_1^{\infty}(G')$

Note that with the relabeling process some information is lost: for example, if G is a labeled aligned subgranularity of H and $G \neq H$, then, after the relabeling, G is not a labeled aligned subgranularity of H. The lost information is semantically meaningful in the calendar algebra, and therefore the relabeling must be performed only when the granularity will not be used as an operator in an algebraic operation.

Let G be a labeled granularity, i and j integers with $i \in \mathcal{L}_G$ s.t. $G(i) \neq \emptyset$. The relabeling operation $Relabel_i^j(G)$ generates a full-integer labeled granularity G' by relabeling $G(i)$ as $G'(j)$ and relabel the next (and previous) granule of G by the next (and previous, respectively) integer. More formally, for each integer k, if $k = j$, then let $G'(k) = G(i)$, and otherwise let $G'(k) = G(i')$ where $G(i')$ is the $|j-k|$-th granule of G after (before, respectively) $G(i)$. If the required

$|j-k|$-th granule of G does not exist, then let $G'(k) = \emptyset$. Note the G' is always a full-integer labeled granularity.

The relabeling procedure can be implemented in the periodic representation we adopted by computing the value of $l_{G'}$. It is easily seen that once $l_{G'}$ is known, the full characterization of G' can be obtained with: $P_{G'} = P_G$; $N_{G'} = R_{G'} = R_G$ and $\overline{\mathcal{L}}_{G'} = \{l_{G'}, l_{G'}+1, \ldots, l_{G'}+N_{G'}-2, l_{G'}+N_{G'}-1\}$. It is clear that the explicit representation of the granules is not modified.

To compute $l_{G'}$ consider the label $i' = i - \left\lfloor \frac{i-l_G}{N_G} \right\rfloor \cdot N_G$; i' represents the label of $\overline{\mathcal{L}}_G$ such that $i - i'$ is a multiple of N_G. Therefore it is clear that the label $j' \in \overline{\mathcal{L}}_{G'}$ s.t. $G'(j') = G(i')$ can be computed by $j' = j - \left\lfloor \frac{i-l_G}{N_G} \right\rfloor \cdot N_{G'}$. Finally $l_{G'}$ is obtained with $l_{G'} = j' - |\delta|$ where δ is the distance, in terms of number of granules of G, from $G(l_G)$ to $G(i')$.

Example 13 *Figure 13 shows an example of the Relabel operation:* $G' = \mathit{Relabel}_{33}^{4}(G)$. *Since* $P_G = 4$ *and* $R_G = 2$ *then* $P_{G'} = 4$ *and* $N_{G'} = 2$. *Moreover,* $i' = 33 - \left\lfloor \frac{33-6}{5} \right\rfloor \cdot 5 = 8$ *and* $j' = 4 - \left\lfloor \frac{33-6}{5} \right\rfloor \cdot 2 = -6$. *Since* $l_G = 6$ *and* $i' = 8$ *then* $G(i')$ *is the next granule of* G *after* $G(l_G)$. *Then* $\delta = 1$ *and hence* $l_{G'} = -6 - 1 = -7$. *It follows that* $\overline{\mathcal{L}}_{G'} = \{-7, -6\}$. *Finally* $G'(-7) = G(6)$ *and* $G'(-6) = G(8)$.

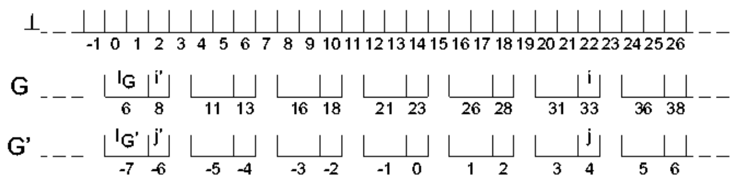


Figure 13: *Relabeling* example

The GSTP constraint solver imposes that the first non-empty granule of any granularity ($\perp$ included) is labeled with 1. Therefore, when using the relabeling operation for producing granularities for GSTP, the parameter j must be set to 1. The parameter i has to be equal to the smallest label among those that identify granules of G covering granules of $\perp$ that are all labeled with positive values. By definition of l_G, $i = l_G$ if $min(\lfloor l_G \rfloor^G) > 0$; otherwise i is the next label of G after l_G.

4.12 Complexity Issues

For each operation the time necessary to perform the three conversion steps, depends on the operation parameters (e.g. the "grouping factor" m, in the *Group* operation) and on the operand granularities (in particular the period length, the period label distance and the number of granules in one period).

A central issue is that if an operand granularity is not the bottom granularity, then its period is a function of the periods of the granularities that are the operands in the operation that defines it. For most of the algebraic operations, in the worst case the period of the resulting granularity is the product of the periods of the operands granularity.

For all operations, the **first step** in the conversion process can be performed in a constant or logarithmic time. Indeed the formulas necessary to derive the period length and the period label distance involve (i) standard arithmetic operations, (ii) the computation of the Greatest Common Divisor and (iii) the computation of the least common multiple. Part (i) can be computed in a constant time while (ii) and (iii) can be computed in a logarithmic time using Euclid's algorithm.

For some operations, the **second step** can be performed in constant time (e.g. *Group*, *Shift* or *Anchored-group*) or in linear time (e.g. set operations). For the other operations it is necessary to compute the set S of labels of a granularity G such that $\forall i \in S\ G(i) \subseteq H(j)$ where H is a granularity and $j \in \mathcal{L}_H$ (analogously if S is the set such that $\forall i \in S\ G(i) \supseteq H(j)$ or $\forall i \in S\ (G(i) \cap H(j) \neq \emptyset)$). This computation needs to be performed once for each granule $i \in P_H^{P_{G'}}$. The idea of the algorithm for solving the problem has been presented in Section 4.2. Several optimizations can be applied to that algorithm, but in the worst case (when H covers the entire time domain) it is necessary to perform a number of $\lceil \cdot \rceil^G$ operations linear in the period length of the resulting granularity. If an optimized data structure is used to represent the granularities, the $\lceil \cdot \rceil^G$ operation can be performed in constant time [4], then the time necessary to perform the second step is linear in the period length of the resulting granularity ($O(P_{G'})$).

The **last step** in the conversion process is performed in linear time with respect to the number of granules in a period of G'.

The complexity analysis of the conversion of a general algebraic expression needs to consider the composition of the operations and hence their complexity. Finally, relabeling, can be done in linear time.

A more detailed complexity analysis is out of the scope of this work.

5 Minimal Representation and Experimental Results

In this section we address the problem of guaranteeing that the converted representation is minimal in terms of the period length. As we will show in Example 14 the conversion formulas proposed in this paper do not guarantee a minimal representation of the result and it is not clear if conversion formulas ensuring minimality exist. Our approach is to apply a minimization step in the conversion.

The practical applicability of the minimization step depends on the period length of the representation that is to be minimized. Indeed, in our tests we noted that the minimization step is efficient if the conversion formulas proposed

[4]If a non-optimized data structure is used, $\lceil \cdot \rceil^G$ requires logarithmic time.

in Section 4 are adopted, while it is impractical when the conversion procedure returns a period that is orders of magnitude higher than the minimal one as would be the case if conversion formulas were constructed in a naive way.

5.1 Period Length Minimization

As stated in Section 2, each granularity can have different periodical representations and, for a given granularity, it is possible to identify a set of representations that are *minimal* i.e. adopting the smallest period length.

Unfortunately, the conversions do not always return a minimal representation, as shown by Example 14.

Example 14 *Consider a calendar that has* `day` *as the bottom granularity. We can define* `week` *as* $\mathtt{week} = \text{Group}_7(\mathtt{day})$*; by applying the formulas for the* Group *operation we obtain* $P_{\mathtt{week}} = 7$ *and* $N_{\mathtt{week}} = 1$*.*

We can now apply the Altering-tick operation to add one day to every first week every two weeks. Let this granularity be $G_1 = \text{Alter}^2_{1,1}(\mathtt{day}, \mathtt{week})$*; applying the formulas for the Altering-tick operation we obtain* $P_{G_1} = 15$ *and* $N_{G_1} = 2$*.*

We can again apply the Altering-tick operation to create a granularity G_2 *by removing one day from every first granule of* G_1 *every two granules of* G_1*:* $G_2 = \text{Alter}^2_{1,-1}(\mathtt{day}, G_1)$*. Intuitively, by applying this operation we should get back to the granularity* `week`*, however using the formulas for the Altering-tick operation we obtain* $P_{G_2} = 14$ *and* $N_{G_2} = 2$*; Hence* G_2 *is not minimal.*

In order to qualitatively evaluate how close to the minimal representations the results of our conversions are, we performed a set of tests using an algorithm (?) for minimality checking. In our experimental results the conversions of algebraic expressions defining granularities in real-world calendars, including many user-defined non-standard ones, always returned exactly minimal representations. Non-minimal ones could only be obtained by artificial examples like the one presented in Example 14.

Although a non-minimal result is unlikely in practical calendars, the minimality of the granularity representation is known to greatly affect the performance of the algorithms for granularity processing, e.g., granularity constraint processing (?), calendar calculations (?), workflow temporal support (?). Hence, we considered an extension of the conversion algorithm by adding a minimization step exploiting the technique illustrated by Bettini et al. (?) to derive a minimal representation.

The choice of using only the conversion algorithm or the extended one with minimizations, should probably be driven by performance considerations. In Section 5.3 we report the results of our experiments showing that generally it is advantageous to apply the minimization step. In our implementation, presented in Section 5.2, it is possible to specify if the minimization step should be performed.

5.2 Implementation of the *CalendarConverter* Web Service

The conversion formulas presented in Section 4 have been implemented into the *CalendarConverter* web service that converts Calendar Algebra representations into the equivalent periodical ones. More precisely, given a calendar in which granularities are expressed by Calendar Algebra operations, the service converts each operation into an equivalent periodical representation.

The service first rewrites each calendar algebra expression in order to express it only in terms of the bottom granularity. For example, if the bottom granularity is `hour`, the expression $\texttt{Monday} = \textit{Select-down}_1^1(\texttt{day}, \texttt{week})$ is changed to

$$\texttt{Monday} = \textit{Select-down}_1^1(\textit{Group}_{24}(\texttt{hour}), \textit{Group}_7(\textit{Group}_{24}(\texttt{hour})))$$

Then, Procedure 1 is run for each granularity's expression. The idea is that the periodical representation of each subexpression is recursively computed starting from the expressions having the bottom granularity as operand. Once each operand of a given operation has been converted to periodical representation, the corresponding formula presented in Section 4 is applied. We call this step the *ConvertOperation* procedure.

A trivial optimization of Procedure 1 consists in caching the results of the conversions of each subexpression so that it is computed only once, even if the subexpression appears several times (like $\textit{Group}_{24}(\texttt{hour})$ in the above `Monday` definition).

Procedure 1 ConvertExpression

- **Input**: a calendar algebra expression *ex*; a boolean value *minimize* that is set to **true** if the minimization step is to be executed;
- **Output**: the periodical representation of *ex*;
- **Method**:

```
if (ex is the bottom granularity) then
    return the periodical representation of the bottom granularity
end if
operands := ∅
for (each operand op of ex) do
    add ConvertExpression(op, minimize) to operands;
end for
result :=ConvertOperation(ex.getOperator(), operands)
if (minimize) then
    minimize the periodical representation of result
end if
return result;
```

Table 1: Impact of the conversion formulas on the performance of the conversion and minimization procedures (time in milliseconds).

Calendar		Section 4 formulas			Less optimized formulas		
Period	Bot	Conv.	Min.	Tot.	Conv.	Min.	Tot.
1 year	day	4	2	6	62	32	94
4 years	day	7	2	9	76	55	131
1 year	hour	9	2	11	2,244	126,904	129,148
4 years	hour	16	4	20	4,362	908,504	912,866
100 years	day	127	9	136	3,764	1,434,524	1,438,288

5.3 Experimental Results

Our experiments address two main issues: first, we evaluate how the conversion formulas impact on the practical applicability of the conversion procedure and, second, we evaluate how useful is the minimization step.

For the first issue, we execute the conversion procedure with two different sets of conversion formulas and compare the results. The first set is laid out in Section 4. The other, that is less optimized, is taken from the preliminary version of this paper (?).

Table 1 shows that when converting calendars having granularities with small minimal period length (first two rows), using the formulas in Section 4 improves the performance by one order of magnitude; However, conversions and minimizations are almost instantaneous with both approaches. On the contrary, when the minimal period length is higher, (last three rows) the time required to minimize the periodical representation is up to five orders of magnitude larger if the formulas proposed by Bettini et al. (?) are used; as a consequence, the entire conversion may require several minutes while, using the formulas presented in Section 4, it still requires only a fraction of a second. If the period length is even larger, the conversion procedure is impractical if the formulas presented by Bettini et al. (?) are used, and indeed in our experiments we did not obtain a result in less than thirteen hours.

For the second issue, we perform a set of three experiments. In the first one we compare the performance of the conversion procedure with the performance of the minimization step. In the experiment we consider the case in which the conversion procedure produces minimal representations. In this case the minimization step is always an overhead since it cannot improve the performance of the conversion procedure.

Figure 14 shows the result of the experiment. Four calendars are considered, each one containing a set of granularities of the Gregorian calendar. The four calendars differs in the values of two parameters: the bottom granularity (it is `second` for cal-1 and cal-3 while it is `minute` for cal-2 and cal-4) and the period in which leap years and leap years exceptions are represented (it is 1, 4, 100 and

400 years for cal-1, cal-3, cal-2 and cal-4 respectively); As a consequence, the minimal period length of the granularities `month` and `year` is about $3 \cdot 10^7$ for cal-1, $5 \cdot 10^7$ for cal-2, 10^8 for cal-3 and $2 \cdot 10^8$ for cal-4.

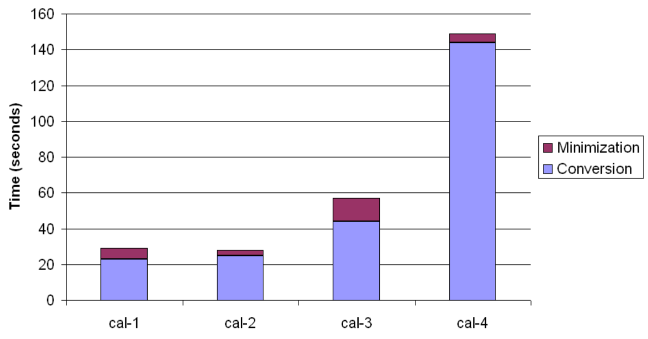


Figure 14: Impact of minimization over conversion; minimal conversions case.

As can be observed in Figure 14, the ratio between the time required to perform the conversions and the time required for the minimization step varies significantly from a minimum of 3% for cal-4 to a maximum of 23% for cal-3. The reason is that the complexity of the conversion procedure is mainly affected by the period length of the granularity having the largest period length. On the other hand, the complexity of the minimization step is affected also by other features of the granularities such as their internal structure and the number of integers that can divide at the same time the period label distance, the period length and the number of granules in one period; For more details see (?).

In the second experiment we consider the case in which the conversion procedure produces a non-minimal representation for a granularity in the input calendar; in this case it is possible to benefit from the minimization step. For example, suppose that a granularity G is converted and that it is then used as an argument of another Calendar Algebra operation that defines a granularity H. The time required to compute the periodical representation of H strongly depends on the period length of G; If the period length of G is reduced by the execution of the minimization step, the conversion of H can be executed faster.

We produced this situation using a technique similar to the one of Example 14; we created Calendar Algebra definitions of the Gregorian calendar in which the granularity `day` is converted into a granularity having a non-minimal representation. Figure 15 shows the performance obtained converting the same granularities that were used in Figure 14. The difference was that in this case the definition of the granularity `day` is such that, after the conversion procedure, its period is twice as large as the minimal one (i.e., 48 hours or 2880 minutes or 172800 seconds depending on the bottom granularity that is used). It can be easily seen that in this case the use of the minimization step can improve the performance of the entire algorithm. Indeed, when the minimization step is performed, the conversion procedure requires about one half of the time that is required when no minimization is performed.

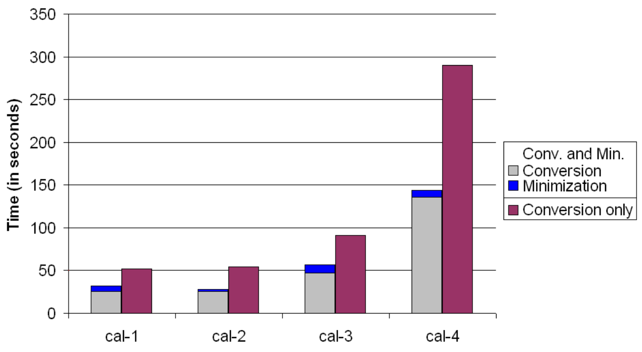


Figure 15: Impact of minimization over conversion; non-minimal case.

In the third experiment we evaluate the impact of the minimal representation on the performance of applications involving intensive manipulations of granularities. In the test we use the GSTP solver as such an application; it computes solutions of temporal constraints with granularities. A description of the architecture of the GSTP system is provided in Section 6.1.

Figure 16 shows our experiments performed on four temporal constraint networks with granularities. The four networks differs in the number of variables, in the number of constraints and in the granularities used to express the constraints. The networks labeled as "non-minimal" use granularities definitions that are obtained with a technique similar to the one used in Example 14, and have a period that is twice as large as the minimal one.

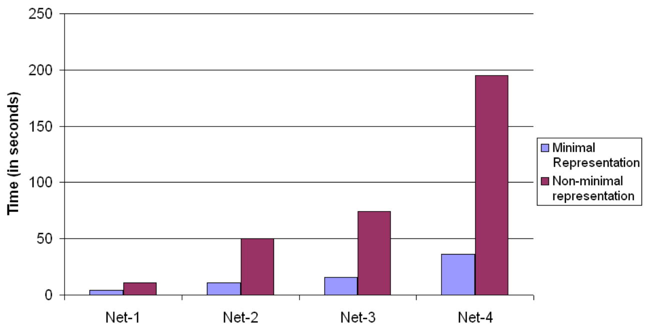


Figure 16: Impact of minimal representations on the performance of the GSTP solver.

Figure 16 shows that the use of minimal representations greatly improves the performance of the GSTP solver. Indeed in our experiments the ratio between the time required to solve the network using a non-minimal representation and a minimal one is between three and five. Moreover, the more time required to solve the network, the greater the improvement obtained using the minimal representation; this means that for very complex temporal networks we expect the improvement to be even higher.

Considering the results of our experiments, we conclude that, in general, it is advisable to perform the minimization step. In particular, it is very advantageous in the specific case of GSTP, based on the following considerations: i) the time required to perform the minimization step is only a fraction of the time required to perform the conversion procedure, ii) the conversions are performed off-line in most cases, with respect to granularity processing, and conversion results are cached for future use, and iii) the period length strongly influences the GSTP processing time that is in most cases much longer than the time needed for conversion.

6 Applications

In this section we complement the motivations for this work with a sketch of the applications enabled by the proposed conversion. Firstly we describe the GSTP system, as an example of applications involving intensive manipulation of time granularities. GSTP is used to check the consistency and to find solutions of temporal constraint satisfaction problems with granularities[5]; It has also been applied to check the consistency of inter-organizational workflow models (?). Then, we discuss the use of Calendar Algebra to define new granularities that may later be part of the input of reasoning services, such as GSTP.

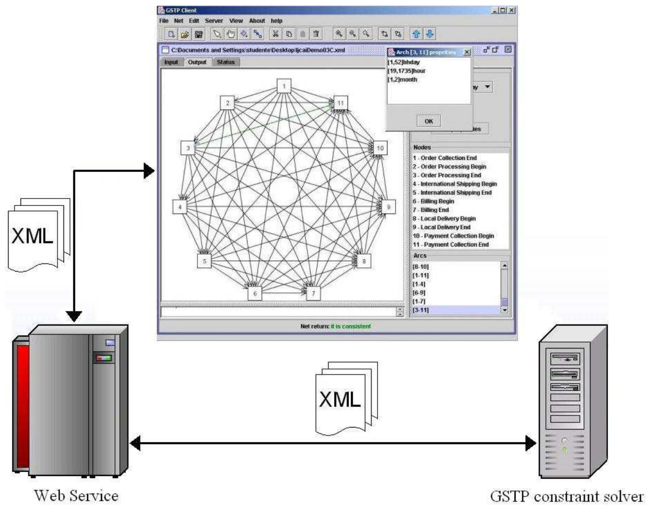
6.1 The GSTP System

The GSTP system has been developed at the University of Milan with the objective of providing universal access to the implementation of a set of algorithms for multi-granularity temporal constraint satisfaction (?). It allows the user to specify binary constraints of the form $Y - X \in [m, n]G$ where m and n are the minimum and maximum values of the distance from Y to X in terms of granularity G. Variables take values in the positive integers, and unary constraints can be applied on their domains. For example, the constraint: *Event2 should occur 2 to 4 business days after the occurrence of Event1* can be modeled by $Occ_{E2} - Occ_{E1} \in [2, 4]BDay$. This problem is considered an extension of STP (?) to multiple and arbitrary granularities. To our knowledge, GSTP is the only available system to solve this class of temporal constraint satisfaction problems.

Figure 17 shows the general architecture of the GSTP system. There are three main modules: the constraint solver; the web service, which enables external access to the solver; and a user interface that can be used locally or remotely to design and analyze constraint networks.

The constraint solver is the C implementation of the ACG algorithm which has been proposed by Bettini et al. (?), and it runs on a server machine. Following the approach of Bettini et al. (?), the solver uses the representation

[5]For a detailed description of the system, see (?).


GSTP Client
File Net Edit Server View About help
C:\Documents and Settings\studente\Desktop\IjcaiDemo03C.xml
Input Output Status
Arch [3, 11] proprities
OK
XML
XML
Web Service
GSTP constraint solver

```
[1,52]bhday
[19,1735]hour
[1,2]month
```

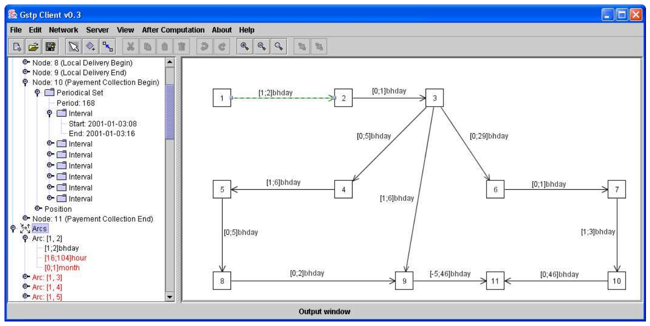
```
Nodes
1 - Order Collection End
2 - Order Processing Begin
3 - Order Processing End
4 - International Shipping Begin
5 - International Shipping End
6 - Billing Begin
7 - Billing End
8 - Local Delivery Begin
9 - Local Delivery End
10 - Payment Collection Begin
11 - Payment Collection End
Arcs
[8-10]
[1-11]
[1-4]
[6-9]
[1-7]
[3-11]
```

```
Net return: it is consistent
```

Figure 17: The GSTP Architecture

of granularities based on periodical sets. This representation makes it possible to efficiently compute the core operations on granularities that are required to solve the constraint satisfaction problem. These operations involve, for example, the union and the intersection of periodical sets. While we cannot exclude that these operations may be computed in terms of alternative low level representations, it seems much harder to obtain similar results if a high level representation, such as Calendar Algebra, is used.

The second module of the system is the Web Service that defines, through a WSDL specification, the parameters that can be passed to the constraint solver, including the XML schema for the constraint network specification.



```
Node: 8 (Local Delivery Begin)
Node: 9 (Local Delivery End)
Node: 10 (Payement Collection Begin)
  Periodical Set
    Period: 168
    Interval
      Start: 2001-01-03:08
      End: 2001-01-03:16
    Interval
    Interval
    Interval
    Interval
    Interval
    Interval
  Position
Node: 11 (Payement Collection End)
Arcs
Arc: [1, 2]
  [1;2]bhday
  [16;104]hour
  [0;1]month
Arc: [1, 3]
Arc: [1, 4]
Arc: [1, 5]
```

Figure 18: The GSTP User Interface

The third module is a remote Java-based user interface, which allows the user to easily edit constraint networks, to submit them to the constraint solver, and to analyze results. In particular, it is possible to have views in terms of specific granularities, to visualize implicit constraints, to browse descriptions of domains, and to obtain a network solution. Fig. 18 shows a screenshot from the interface.

6.2 Defining New Granularities

While the GSTP solver can handle arbitrary granularities, new granularities must be added by editing their explicit periodical representation. This is true in general for any multi-granularity reasoning service based on a low-level representation of granularities, and it is a painful task when the granularities have a large period. For example, in the experimental results illustrated in Figure 16, we used a representation of the granularity `month` that considers leap years and leap years exceptions in a period of 400 years. In this case, the users have to specify the representation of 4800 granules i.e., the number of months in 400 years.

Because the period length of real world granularities is generally high, a graphical interface does not help if it only supports the user to individually select the explicit granules. An effective solution requires the use of implicit or explicit operations on granules. Among the various proposals, Calendar Algebra provides the richest set of such operators. A question arises: is the definition of granularities in terms of Calendar Algebra really simpler than the specification of the periodical representation? Calendar Algebra does not seem to be user friendly: the exact semantics of each operator may not be immediate for an inexperienced user and some time is required in order to learn how to use each operator.

In practice, we do not think that it is reasonable to ask an unexperienced user to define granularities by writing Calendar Algebra expressions. Nevertheless, we do think that Calendar Algebra can be used by specialized user interfaces to guide the user when specifying granularities. In this sense, we believe that Calendar Algebra plays the same role that SQL does in the definition of databases queries. Similarly to Calendar Algebra, SQL is an abstraction tool that can be directly exploited in all its expressive power by an advanced user, but can also be used by a less experienced user through a graphical user interface, possibly with a reduced expressiveness.

As mentioned above, in the case of periodical representations, graphical user interfaces are not sufficient for making the specification of new granularities practical. On the contrary, in the case of Calendar Algebra, user interfaces can strongly enhance the usability of Calendar Algebra, making its practical use possible also for the definition of involved granularities. There are at least two reasons for this difference. Firstly, the main difficulty of Calendar Algebra is the understanding of the semantics of the operators and the choice of the most appropriate one for a given task. An effective user interface can hide the existence of the algebraic operators to the user showing only how the operators modify existing granularities (i.e., the semantics of the operators). Secondarily, Calendar Algebra allows the compact definition of granularities. This is due to the fact that the Calendar Algebra operations are specifically designed to reflect the intuitive ways in which users define new granularities.

Example 15 shows how a graphical user interface can be effectively used to define a new granularity in terms of Calendar Algebra expression.

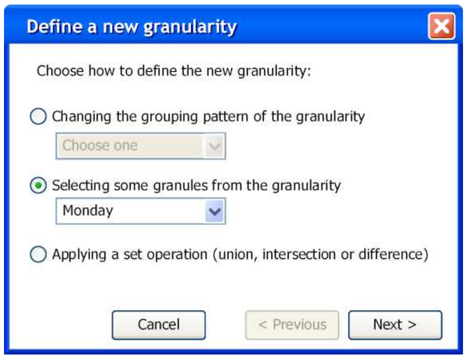
Example 15 *This example shows how a graphical user interface can be used to support the user in the definition of the granularity* `final` *as the set of days, each one corresponding to the last Monday of every academic semester. We assume that the granularities* `Monday` *and* `academicSemester` *have already been defined. The graphical user interface that we use in this example is a wizard that guides the user step by step. In the first step (Figure 19(a)) the user chooses the kind of operation he wants to perform. In the second step (Figure 19(b)) the user can provide more details about how he wants to modify the operand granularity (*`Monday`*, in the example). The results of this choice is a Calendar Algebra expression that is shown in the third step (Figure 19(c)); in this last window the user can also give a name to the granularity that has been defined.*

6.3 The Global Architecture

Figure 20 shows a possible architecture for the integration of GSTP, the interface for new granularity definitions and the CalendarConverter web service. A granularity repository collects the Calendar Algebra definitions. Upon request by the GSTP system definitions are converted in low-level representation by the CalendarConverter web service to be efficiently processed. Clearly, caching techniques can be used to optimize the process.

7 Related Work

Several formalisms have been proposed for symbolic representation of granularities and periodicity. Periodicity and its application in the AI and DB area have been extensively investigated (?, ?, ?, ?). Regarding symbolic representation, it is well known the formalism proposed by Leban et al. (?), that is based on the notion of *collection*, and it is intended to represent temporal expressions occurring in natural language. A *collection* is a structured set of time intervals where the order of the collection gives a measure of the structure depth: an order 1 collection is an ordered list of intervals, and an order n ($n > 1$) collection is an ordered list of collections having order $n - 1$. Two operators, called *slicing* and *dicing* are used to operate on collections by selecting specific intervals or sub-collections, and by further dividing an interval into a collection, respectively. For example, `Weeks:during:January2006` divides the interval corresponding to `January2006` into the intervals corresponding to the weeks that are fully contained in that month. This formalism has been adopted with some extensions by many researchers in the AI (?, ?) and Database area (?, ?). In particular, the control statements `if-then-else` and `while` have been introduced by Chandra


Define a new granularity

Choose how to define the new granularity:

Changing the grouping pattern of the granularity

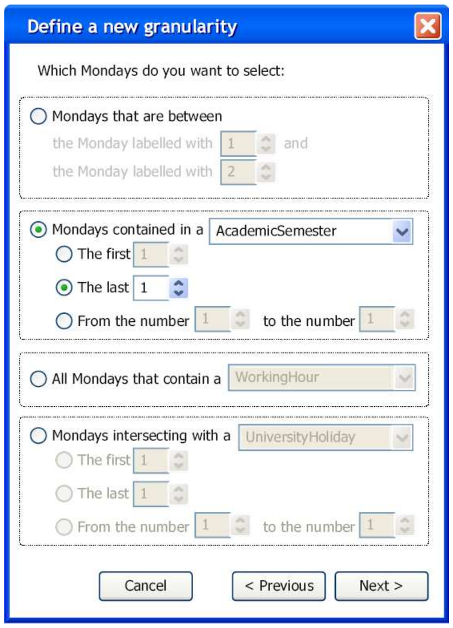
Choose one

[✓ Selecting some granules from the granularity]

Monday

Applying a set operation (union, intersection or difference)

(a) Step 1.


Define a new granularity

Which Mondays do you want to select:

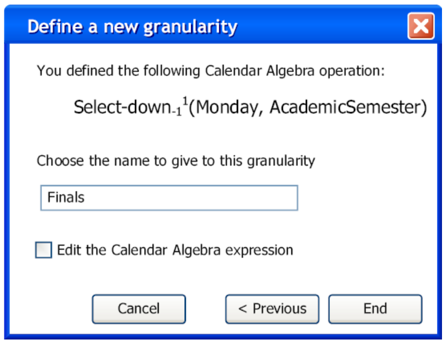
Mondays that are between
the Monday labelled with 1 and
the Monday labelled with 2

[✓ Mondays contained in a] AcademicSemester
The first 1
[✓ The last] 1
From the number 1 to the number 1

All Mondays that contain a WorkingHour

Mondays intersecting with a UniversityHoliday
The first 1
The last 1
From the number 1 to the number 1

(b) Step 2.


Define a new granularity

You defined the following Calendar Algebra operation:

Select-down$_{-1}^{1}$(Monday, AcademicSemester)

Choose the name to give to this granularity

Finals

☐ Edit the Calendar Algebra expression

(c) Step 3.

Figure 19: A 3-steps wizard for visually defining a granularity using Calendar Algebra

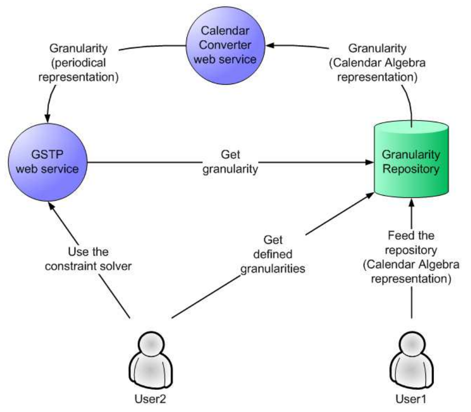

Calendar Converter web service
Granularity (periodical representation)
Granularity (Calendar Algebra representation)
GSTP web service
Get granularity
Granularity Repository
Use the constraint solver
Get defined granularities
Feed the repository (Calendar Algebra representation)
User2
User1

Figure 20: Integration of GSTP and CalendarConverter web services

et al. (?) to facilitate the representation of certain sets of intervals. For example, it is possible to specify: *the fourth Saturday of April if not an holiday, and the previous business day otherwise.*

As for the deductive database community, a second influential proposal is the *slice* formalism introduced by Niezette et al. (?). A slice denotes a (finite or infinite) set of not necessarily consecutive time intervals. For example, the slice `all.Years + {2,4}.Months + {1}.Days ▷ 2.Days` denotes a set of intervals corresponding to the first 2 days of February and April of each year.

A totally different approach is the *calendar algebra* described by Ning et al. (?), and considered in this paper. The representation is based on a rich set of algebraic operators on periodic sets as opposed to *slicing* and *dicing* over nonconvex intervals.

None of the above cited papers provide a mapping to identify how each operator changes the mathematical characterization of the periodicity of the argument expressions. The problem of finding these mappings is not trivial for some operators.

In (?) the expressive power of the algebras proposed by Leban et al. (?) and Niezette et al. (?) is compared and an extension to the first is proposed in order to capture a larger set of granularities. Since the periodical representation is used to compare expressiveness, a mapping from calendar expressions in those formalisms to periodical representations can be found in the proofs of that paper. However, since minimality is not an issue for the purpose of comparing expressiveness, in many cases the mapping returns non-minimal representations.

Regarding alternative approaches for low-level representation, we already mentioned that the ones based on strings (?) and automata (?, ?) may be considered as an alternative for the target of our conversion. As a matter of fact, an example of the conversion of a Calendar Algebra expression into a string based representation can be found in (?). A complete conversion procedure appeared during the revision process of this paper in the PhD Dissertation by Puppis

(?). The aim of the conversion is to prove that the *granspecs* formalism, used to represent granularities in terms of automata, has at least the same expressiveness as the Calendar Algebra. Hence, obtaining minimal representations was not the goal. Moreover, in their case minimization is not in terms of the period length, but in terms of the automaton size and automaton complexity. About the complexity of reasoning, given an automaton M, the worst case time complexity of the operations analogous to our *up* and *down* depends linearly on $||M||$, a value computed from M itself and called *complexity of* M. In this sense $||M||$ has the same role of our period length (P), even if a precise relationship between the two values is hard to obtain. In our approach we compute *up* in logarithmic time with respect to P and *down* in linear time with respect to the dimension of the result (that is bounded by P). Other operations, like checking for equivalence, seem to be more complex using automata (?). Techniques for minimization in terms of automaton complexity are presented by Dal Lago et al. (?), and the time complexity is proved to be polynomial, even if the exact bound is not explicitly given. In our approach, the worst case time complexity for the minimization is $O(P^{\frac{3}{2}})$ (?). Overall, the automata approach is very elegant and well-founded, but, on one side it still misses an implementation in order to have some experimental data to compare with, and on the other side only basic operations have been currently defined; it would be interesting to investigate the definition on that formalism of more complex operations like the ones required by GSTP.

8 Conclusion and Future Work

We have presented an hybrid algorithm that interleaves the conversion of Calendar Algebra subexpressions into periodical sets with the minimization of the period length. We have proved that the algorithm returns set-based granularity representations having minimal period length, which is extremely important for the efficiency of operations on granularities. Based on the technical contribution of this paper, a software system is being developed allowing users to access multi-granularity reasoning services by defining arbitrary time granularities with a high-level formalism. Our current efforts are mainly devoted to completing and refining the development of the different modules of the architecture shown in Section 6.3.

As a future work, we intend to develop effective graphical user interfaces to support the definition of Calendar Algebra expressions in a user friendly way. Example 15 described one of the possible interfaces. Another open issue is how to convert a periodical representation of a granularity into a "user friendly" Calendar Algebra expression. This conversion could be useful, for example, to present the result of a computation performed using the periodical representation. However, a naive conversion may not be effective since the resulting calendar algebra expression could be as involved as the periodical representation from which it is derived. For example, a conversion procedure is presented by Bettini et al. (?) to prove that the Calendar Algebra is at least as expressive as the periodical representation; however, the resulting Calendar Algebra expression is composed by a number of Calendar Algebra operations that is linear in the number of granules that are in one period of the original granularity. On the contrary, an effective conversion should generate Calendar Algebra expressions that are compact and easily readable by the user. This problem is somehow related to the discovery of calendar-based association rules (?). Finally, we intend to investigate the usage of the automaton-based representation as a low-level granularity formalism. It would be interesting to know whether, using this representation, it is possible to compute the same operations that can be computed with the periodical representation and if any performance gain could be achieved.

Acknowledgments

We thank the anonymous referees for their useful comments and suggestions. The work of Bettini and Mascetti was partially supported by Italian MIUR InterLink project N.II04C0EC1D. The work of Wang was partially supported by the US NSF grant IIS-0415023.

A Proofs

A.1 Transitivity of the *Periodically Groups Into* Relationship

In order to prove the correctness of the conversions of algebraic expressions into periodical sets, it is useful to have a formal result about the transitivity of the *periodically groups into* relation. In addition to transitivity of $\trianglelefteq$, Theorem 1 also says something about period length values.

Theorem 1 *Let G and H be two unbounded granularities such that G is periodic in terms of the bottom granularity (i.e., $\perp \trianglelefteq G$) and H is periodic in terms of G (i.e., $G \trianglelefteq H$). Let P_H^G and N_H^G be the period length and the period label distance of H in terms of granules of G, and N_G the period label distance of G in terms of $\perp$. Then, if $P_H^G = \alpha N_G$ for some positive integer α, then H is periodic in terms of the bottom granularity (i.e., $\perp \trianglelefteq H$) and $P_H = \alpha P_G$.*

Proof. Since by hypothesis $G \trianglelefteq H$ and $P_H^G = \alpha N_G$, $\forall i$ if $H(i) = \bigcup_{r=0}^{n_i} G(i_r)$, then $H(i + N_H^G) = \bigcup_{r=0}^{n_i} G(i_r + \alpha N_G)$. This can be also written as follows:
if

$$H(i) = G(i_0) \cup ... \cup G(i_{n_i}) \tag{1}$$

then $\exists \beta \in \mathbb{N} s.t.$:

$$H(i + N_H^G) = G(i_0 + \alpha N_G) \cup ... \cup G(i_{n_i} + \alpha N_G) \tag{2}$$

Since $\perp \trianglelefteq G$, if

$$G(i_j) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k}) \tag{3}$$

then

$$G(i_j + N_G) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k} + P_G) \tag{4}$$

This can be clearly extended using αN_G instead of N_G.

$$G(i_j + \alpha N_G) = \bigcup_{k=0}^{\tau_{i_j}} \perp(i_{j,k} + \alpha P_G) \tag{5}$$

Rewriting (1) substituting $G(i_j)$ according to (3) and rewriting (2) substituting $G(i_j + \alpha N_G)$ according to (5), we obtain:
if $H(i) = \underbrace{\perp(i_{0,0}) \cup \ldots \cup \perp(i_{0,\tau_{i_0}})}_{G(i_0)} \cup \ldots \cup \underbrace{\perp(i_{n_i,0}) \cup ... \cup \perp(i_{n_i,\tau_{i_{n_i}}})}_{G(i_{n_i})}$

then $H(i+N_H^G) = \underbrace{\perp(i_{0,0}+\alpha P_G)\cup\ldots\cup\perp(i_{0,\tau_{i_0}}+\alpha P_G)}_{G(i_0+\alpha N_G)}\cup\ldots$
$\cup\underbrace{\perp(i_{n_i,0}+\alpha P_G)\cup\ldots\cup\perp(i_{n_i,\tau_{in_i}}+\alpha P_G)}_{G(i_{n_i}+\alpha N_G)}$

Hence the second condition of Definition 5 is satisfied. The third one is always satisfied for unbounded granularities. The first one is satisfied too; in fact since $G \trianglelefteq H$ with a period label distance of N_H^G, then for each label i of H, $i+N_H^G$ is a label of H. Hence, by definition of periodically-groups-into $\perp \trianglelefteq H$ with $P_H = \alpha P_G$ and $N_H = N_H^G$. □

A.2 Proof of Proposition 1

A.2.1 Part 1

From the definition of the *Group* operation, for all $i \in \mathbb{N}$:

$$G'(i) = \bigcup_{j=(i-1)m+1}^{im} G(j) = G(im-m+1)\cup\ldots\cup G(im) = G(\lambda)\cup\ldots\cup G(\lambda+m-1)$$

with $\lambda = im-m+1$. Furthermore, $\forall k \in \mathbb{N}$:

$$G'(i+k) = \bigcup_{j=(i+k-1)m+1}^{(i+k)m} G(j) = G(im+km-m+1)\cup\ldots\cup G(im+km) =$$
$$= G(\lambda+km)\cup\ldots\cup G(\lambda+km+m-1)$$

Hence,

$$\text{If } G'(i') = \bigcup_{r=0}^{m-1} G(\lambda+r) \text{ then } G'(i'+k) = \bigcup_{r=0}^{m-1} G(\lambda+r+km). \tag{6}$$

This holds for each k. If we use $k = \frac{N_G}{GCM(m,N_G)}$ (note that $k \in \mathbb{N}$), then all the hypotheses of Theorem 1 are satisfied: (i) $\perp \trianglelefteq G$ (by hypothesis); (ii) $G \trianglelefteq G'$ (since $G \trianglelefteq G'$, $\mathcal{L}_{G'} = \mathbb{Z}$, and (6) holds); (iii) $P_{G'}^G = \frac{m\cdot N_G}{GCM(m,N_G)}$ (since we use $k = \frac{N_G}{GCM(m,N_G)}$ and, from (6) we know that $P_{G'}^G = km$). Therefore, by Theorem 1, $\perp \trianglelefteq G'$ with $P_{G'} = \frac{mP_G}{GCM(m,N_G)}$ and $N_{G'} = \frac{N_G}{GCM(m\cdot N_G)}$.

A.2.2 Part 2

By definition of l, we need to show that $G'\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right) = \bigcup_{j=b}^{t} G(j)$ with $b \le l_G \le t$.

From the definition of the *Group* operation, $G'(i) = \bigcup_{j=(i-1)\cdot m+1}^{i\cdot m} G(i)$; hence:

$$G'\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right) = \bigcup_{j=\left\lfloor\frac{l_G-1}{m}\right\rfloor\cdot m+1}^{\left(\left\lfloor\frac{l_G-1}{m}\right\rfloor+1\right)\cdot m} G(j)$$

We prove the thesis showing that (1) $\left\lfloor \frac{l_G-1}{m} \right\rfloor \cdot m+1 \leq l_G$ and that (2) $\left(\left\lfloor \frac{l_G-1}{m} \right\rfloor +1\right) \cdot m \geq l_G$.

(1) Since $\left\lfloor \frac{l_G-1}{m} \right\rfloor \leq \frac{l_G-1}{m}$, hence $\left\lfloor \frac{l_G-1}{m} \right\rfloor \cdot m+1 \leq l_G$

(2) First we prove that $\left\lfloor \frac{l_G-1}{m} \right\rfloor \geq \frac{l_G}{m}-1$. Since $\left\lfloor \frac{l_G-1}{m} \right\rfloor = \frac{l_G-1-[(l_G-1) mod\ m]}{m}$ we have to prove that $\frac{l_G-1-[(l_G-1) mod\ m]}{m} \geq \frac{l_G}{m}-1$; it is equivalent to the inequality $-\left[(l_G-1) mod\ m\right] \geq -m+1$ that is true since $(l_G-1) mod\ m \leq m-1$. Since $\left\lfloor \frac{l_G-1}{m} \right\rfloor \geq \frac{l_G}{m}-1$ it is trivial that $\left(\left\lfloor \frac{l_G-1}{m} \right\rfloor +1\right) \cdot m \geq l_G$.

A.3 Proof of Proposition 2

A.3.1 Part 1

Proof sketch

We show that $G_2 \trianglelefteq G'$ with $P_{G'}^{G_2} = \alpha N_{G_2}$ and then we apply Theorem 1 to obtain the thesis. In particular we use

$$\Delta = lcm\left(N_{G_1}, m, \frac{P_{G_2} \cdot N_{G_1}}{GCD(P_{G_2} \cdot N_{G_1}, P_{G_1})}, \frac{N_{G_2} \cdot m}{GCD(N_{G_2} \cdot m, |k|)}\right)$$

and

$$\alpha = \left(\frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}\right) \cdot \frac{P_{G_2}}{N_{G_2}}$$

such that, for each i, if $\exists j,k : G'(i) = \bigcup_{r=0}^{k} G_2(j+r)$, then $G'(i+\Delta) = \bigcup_{r=0}^{k} G_2(j+r+\alpha N_{G_2})$.

Given an arbitrary granule $G'(i)$, we show that $G'(i+\Delta)$ is the union of granules that can be obtained by adding αN_{G_2} to the index of each granule of G_2 contained in $G'(i)$. Note that $i+\Delta \in \mathcal{L}_{G'}$ since G' is full-integer labeled. In order to show that this is correct we consider the way granules of G' are constructed by definition of altering-tick. More precisely, we compute the difference between the label $b'_{i+\Delta}$ of the first granule of G_2 included in $G'(i+\Delta)$ and the label b'_i of the first granule of G_2 included in $G'(i)$; we show that this difference is equal to the difference between the label $t'_{i+\Delta}$ of the last granule of G_2 included in $G'(i+\Delta)$ and the label t'_i of the last granule of G_2 included in $G'(i)$. This fact together with the consideration that G_2 is a full-integer labeled granularity, leads to the conclusion that $G'(i)$ and $G'(i+\Delta)$ have the same number of granules. It is then clear that the above computed label differences are also equal to the difference between the label of an arbitrary n-th granule of G_2 included in $G'(i+\Delta)$ and the label of the n-th granule of G_2 included in $G'(i)$. If this difference is $b'_{i+\Delta} - b'_i$, then we have: if $\exists j,k : G'(i) = \bigcup_{r=0}^{k} G_2(j+r)$, then $G'(i+\Delta) = \bigcup_{r=0}^{k} G_2\left(j+r+\left(b'_{i+\Delta}-b'_i\right)\right)$. By showing that $b'_{i+\Delta}-b'_i$ is a multiple of N_{G_2} the thesis follows.

Proof details

Assume $G_1(i) = \bigcup_{j=b_i}^{t_i} G_2(j)$ and $G_1(i+\Delta) = \bigcup_{j=b_{i+\Delta}}^{t_{i+\Delta}} G_2(j)$. We need to compute $b'_{i+\Delta} - b'_i$. From the definition of the the altering-tick operation:

$$b'_i = \begin{cases} b_i + \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) k & \text{if } i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l, \\ b_i + \left(\left\lfloor \frac{i-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{7}$$

and

$$b'_{i+\Delta} = \begin{cases} b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) k & \text{if } i+\Delta = \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l, \\ b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{8}$$

Note that if $i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l$, then $i + \Delta = \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l$. Indeed, $\left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) m + l = \left(\left\lfloor \frac{i-l}{m} + \frac{\Delta}{m} \right\rfloor\right) m + l$ and, since Δ is a multiple of m, then $\left(\left\lfloor \frac{i-l}{m} + \frac{\Delta}{m} \right\rfloor\right) m + l = \left(\frac{\Delta}{m} + \left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l = \Delta + \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l$.

Hence, to compute $b'_{i+\Delta} - b'_i$ we should consider two cases:

$$b'_{i+\Delta} - b'_i = \begin{cases} b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor\right) k - b_i - \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) k & \text{if } i = \left(\left\lfloor \frac{i-l}{m} \right\rfloor\right) m + l \\ b_{i+\Delta} + \left(\left\lfloor \frac{i+\Delta-l}{m} \right\rfloor + 1\right) k - b_i - \left(\left\lfloor \frac{i-l}{m} \right\rfloor + 1\right) k & \text{otherwise.} \end{cases} \tag{9}$$

In both cases (again considering the fact that Δ is a multiple of m):

$$b'_{i+\Delta} - b'_i = (b_{i+\Delta} - b_i) + \frac{\Delta \cdot k}{m} \tag{10}$$

We are left to compute $b_{i+\Delta} - b_i$, i.e., the distance in terms of granules of G_2, between $G_2(b_i)$ and $G_2(b_{i+\Delta})$. Since, by hypothesis, $G_1(i) = \bigcup_{j=b_i}^{t_i} G_2(j)$ and $G_1(i+\Delta) = \bigcup_{j=b_{i+\Delta}}^{t_{i+\Delta}} G_2(j)$, then the first granule of $\perp$ making $G_2(b_i)$ and the first granule of $\perp$ making $G_1(i)$ is the same granule. The same can be observed for the first granule of $\perp$ making $G_2(b_{i+\Delta})$ and the first granule of $\perp$ making $G_1(i+\Delta)$. More formally:

$$min\, \lfloor b_i \rfloor^{G_2} = min\, \lfloor i \rfloor^{G_1}$$

and

$$min\, \lfloor b_{i+\Delta} \rfloor^{G_2} = min\, \lfloor i+\Delta \rfloor^{G_1}$$

Hence, we have:

$$min\, \lfloor b_{i+\Delta} \rfloor^{G_2} - min\, \lfloor b_i \rfloor^{G_2} = min\, \lfloor i+\Delta \rfloor^{G_1} - min\, \lfloor i \rfloor^{G_1} \tag{11}$$

We have shown that the difference between the index of the first granule of $\perp$ making $G_2(b_{i+\Delta})$ and the index of the first granule of $\perp$ making $G_2(b_i)$ is equal to the difference between the index of the first granule of $\perp$ making $G_1(i+\Delta)$ and the index of the first granule of $\perp$ making $G_1(i)$. Then, we need to compute the difference between the index of the first granule of $\perp$ making $G_1(i+\Delta)$ and the index of the first granule of $\perp$ making $G_1(i)$. Since $\perp \trianglelefteq G_1$

and Δ is a multiple of N_{G_1}, for each i, if $\exists j, \tau : G_1(i) = \bigcup_{r=0}^{\tau} \perp(j+r)$, then $G_1(i+\Delta) = \bigcup_{r=0}^{\tau} \perp(j + \frac{\Delta \cdot P_{G_1}}{N_{G_1}})$. Hence, this difference has value $\frac{\Delta \cdot P_{G_1}}{N_{G_1}}$, and for what shown above this is also the value of the difference between the index of the first granule of $\perp$ making $G_2(b_{i+\Delta})$ and the index of the first granule of $\perp$ making $G_2(b_i)$. Then, since $\perp \trianglelefteq G_2$ with period length P_{G_2} and since $\frac{\Delta \cdot P_{G_1}}{N_{G_1}}$ is a multiple of P_{G_2}, we have that, if:

$$\perp(j) \subseteq G_2(i)$$

then:

$$\perp(j + \frac{\Delta \cdot P_{G_1}}{N_{G_1}}) \subseteq G_2(i + \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}})$$

Thus, $b_{i+\Delta} - b_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}}$.

Reconsidering 10:

$$b'_{i+\Delta} - b'_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}.$$

Analogously we can compute $t'_{i+\Delta} - t'_i = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}} + \frac{\Delta \cdot k}{m}$.

Thus, $b'_{i+\Delta} - b'_i = t'_{i+\Delta} - t'_i$; hence $t_{i+\Delta} - b_{i+\Delta} = t_i - b_i$. Since G_2 is a full integer labeled granularity, then $G'(i)$ and $G'(i+\Delta)$ are formed by the same number of granules.

Since we now know $G'(i+\Delta) = \bigcup_{j=b'_{i+\Delta}}^{t'_{i+\Delta}} G_2(j) = \bigcup_{j=b'_i}^{t'_i} G_2(j + (b'_{i+\Delta} - b'_i))$ and $(b'_{i+\Delta} - b'_i)$ is a multiple of N_{G_2}, we have $G_2 \trianglelefteq G'$, $P_{G'}^{G_2} = \frac{\Delta \cdot P_{G_1} \cdot N_{G_2}}{N_{G_1} \cdot P_{G_2}}$ and $\perp \trianglelefteq G_2$. Hence, all the hypothesis of Theorem 1 hold, and its application leads the thesis of this proposition.

A.3.2 Part 2

Since G_2 partitions G' (see table 2.2 of (?)), then (1) $\lceil l_{G_2} \rceil_{G_2}^{G'}$ is always defined and (2) $min(\{n \in \mathbb{N}^+ | \exists i \in \mathcal{L}_{G_2} s.t.\ \perp(n) \subseteq G_2(i)\}) = min(\{m \in \mathbb{N}^+ | \exists j \in \mathcal{L}_{G'} s.t.\ \perp(m) \subseteq G'(j)\})$. Therefore $l_{G'}$ is the label of the granule of G' that covers the granule of G_2 labeled with l_{G_2}; by definition of $\lceil \cdot \rceil$ operation, $l_{G'} = \lceil l_{G_2} \rceil_{G_2}^{G'}$.

A.4 Proof of Proposition 3

A.4.1 Part 2

By definition of the *Shift* operation, $G'(i) = G(i-m)$. Hence $G'(l_G + m) = G(l_G + m - m) = G(l_G)$.

A.5 Proof of Proposition 4

A.5.1 Part 1

The thesis will follow from the application of Theorem 1. Indeed, we know that $\perp \trianglelefteq G_2$ and we show that $G_2 \trianglelefteq G'$ with $P_{G'}^{G_2}$ multiple of N_{G_2}. For this we need to identify Δ and α s.t., for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j\in s(i)} G_2(j)$, then $G'(i+\Delta) = \bigcup_{j\in s(i)} G_2(j+\alpha N_{G_2})$.

Consider an arbitrary $i \in \mathbb{N}$ and $\Delta = \frac{lcm(P_{G_1},P_{G_2})N_{G_1}}{P_{G_1}}$. By definition of the combining operation, we have $G'(i) = \bigcup_{j\in s(i)} G_2(j)$ and $G'(i+\Delta) = \bigcup_{j\in s(i+\Delta)} G_2(j)$ with

$$s(i) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$$

and

$$s(i+\Delta) = \{j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i+\Delta)\}.$$

We now show that $s(i+\Delta)$ is composed by all and only the elements of $s(i)$ when the quantity $\Delta' = \frac{lcm(P_{G_1},P_{G_2})N_{G_2}}{P_{G_2}}$ is added. For this purpose we need:

$$\forall j \in s(i)\ \exists (j+\Delta') \in s(i+\Delta) \tag{12}$$

and

$$\forall\, (j+\Delta') \in s(i+\Delta)\ \exists j \in s(i) \tag{13}$$

About 12, note that if $j \in s(i)$, then $G_2(j) \subseteq G_1(i)$. Since $\perp \trianglelefteq G_2$, if

$$G_2(j) = \bigcup_{r=0}^{k} \perp(j_r)$$

then

$$G_2\,(j+\Delta') = \bigcup_{r=0}^{k} \perp(j_r + lcm(P_{G_1}, P_{G_2})) \tag{14}$$

Since $G_1(i) \supseteq G_2(j) = \bigcup_{r=0}^{k} \perp(j_r)$, and since $\perp \trianglelefteq G_1$, then

$$G_1(j+\Delta) \supseteq \bigcup_{r=0}^{k} \perp(j_r + lcm(P_{G_1}, P_{G_2})) \tag{15}$$

From 14 and 15 we derive $G_1(i+\Delta) \supseteq G_2(j+\Delta')$, and hence $(j+\Delta') \in s(i+\Delta)$. Analogously can be proved the validity of 13; Hence, for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j\in s(i)} G_2(j)$, then $G'(i+\Delta) = \bigcup_{j\in s(i)} G_2(j+\Delta')$. Hence, considering the fact that $G_2 \trianglelefteq G'$, we can conclude $G_2 \trianglelefteq G'$. Finally, since $P_{G'}^{G_2}$ is a multiple of N_{G_2}, by Theorem 1 we obtain the thesis.

A.5.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \widetilde{s}(i) \neq \emptyset\}$$

where $\forall i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}\ \widetilde{s}(i) = \{j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | \emptyset \neq G_2(j) \subseteq G_1(i)\}$;

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that: (1) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that exists $k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$ and since G' is derived by the *Combine* operation, then $\exists q \in \mathcal{L}_{G_2} | G_2(q) \subseteq G_1(k)$. By definition of the *Combine* operation $G'(k) = \bigcup_{j \in s(k)} G_2(j)$; since $q \in s(k)$, then $G_2(q) \subseteq G'(k)$. Hence (a) $\exists q \in \mathcal{L}_{G_2} | G_2(q) \subseteq G'(k)$.

Moreover, since $k \notin \widetilde{\mathcal{L}}_{G'}$, then $\widetilde{s}(k) = \emptyset$; therefore $\nexists \mathbf{j} \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(k)$. By definition of the *Combine* operation it is easily seen that $G' \preceq G_1$. Using this and the previous formula, we derive that (b) $\nexists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G'(k)$.

From (a) and (b) it follows that $\exists q \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(q) \subseteq G'(k)$. We show that this leads to a contradiction.

Since $q \notin \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ and labels of $\hat{\mathcal{L}}_{G_2}^{P_{G'}}$ are contiguous (i.e., $\nexists i \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ s.t. $min(\hat{\mathcal{L}}_{G_2}^{P_{G'}}) < i < max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$), then $q < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ or $q > max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$. We consider the first case, the proof for the second is analogous.

If $q < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ then $max(\lfloor q \rfloor^{G_2}) < 1$ (otherwise $q \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$).

Let be $\alpha = min(\lfloor min(\hat{\mathcal{L}}_{G'}) \rfloor^{G'})$. Since $k \in \hat{\mathcal{L}}_{G'}$, then $\alpha \leq \lfloor k \rfloor^{G'}$.

If $\alpha \geq 1$, then $G'(k) \cap G_2(q) = \emptyset$ contradicting $G'(k) \supseteq G_2(q)$.

If $\alpha < 1$, then $G'(l_{G'}) \supseteq \perp(0)$ and we show that $l_{G'} \in \widetilde{\mathcal{L}}_{G'}$. Indeed, by definition of *Combine*, $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G'(L_{G'})$. Since $G' \preceq G_1$ we also have $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(L_{G'})$; hence $j \in \widetilde{s}(l_{G'})$ and then $l_{G'} \in \widetilde{\mathcal{L}}_{G'}$.

Since $0 \in G'(l_{G'})$ and $max(\lfloor q \rfloor^{G_2}) \leq 0$, then $max(\lfloor q \rfloor^{G_2}) < \alpha$ (otherwise $G_2(q) \subseteq G'(l_{G'})$). Therefore, since $min(\lfloor k \rfloor^{G'}) \geq \alpha$, then $\lfloor q \rfloor^{G_2} \cap \lfloor l_{G'} \rfloor^{G'} = \emptyset$, in contradiction with $G_2(q) \subseteq G'(k)$.

(2) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$, by definition of $\widetilde{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ and $\widetilde{s}(k) \neq \emptyset$; Therefore, by definition of $\widetilde{s}$, $\exists j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}} | G_2(j) \subseteq G_1(k)$.

Since $j \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$, by definition of $\hat{\mathcal{L}}$, $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_2} = j$. Since $G_2(j) \subseteq G_1(k)$, then $\lceil h \rceil^{G_1} = k$. By definition of the *combine* operation, $\lceil h \rceil^{G'} = k$. Moreover, since $0 < h \leq P_{G'}$, by definition of $\hat{\mathcal{L}}$, $\lceil h \rceil^{G'} = k \in \hat{\mathcal{L}}_{G'}$, contradicting the hypothesis.

A.6 Proof of Proposition 5

A.6.1 Part 1

The thesis will follow from the application of Theorem 1. Indeed, we show that $G_1 \trianglelefteq G'$ with $P_{G'}^{G_1}$ multiple of N_{G_1}. For this we need to identify Δ and α s.t., for each i, if there exists $s(i)$ s.t. $G'(i) = \bigcup_{j \in s(i)} G_1(j)$, then $G'(i + \Delta) = \bigcup_{j \in s(i)} G_1(j + \alpha N_{G_1})$. Let $\Delta = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$. By definition of anchored grouping, $G'(i) = \bigcup_{j=i}^{i'-1} G_1(j)$ and $G'(i+\Delta) = \bigcup_{j=i+\Delta}^{(i+\Delta)'-1} G_1(j)$ where i' is the first label of G_2 after i and $(i+\Delta)'$ is the first label of G_2 after $i+\Delta$. By periodicity of G_2, (and since Δ is a multiple of N_{G_2}) the difference between the label of the granule following $G_2(i+\Delta)$ and the label of the granule following $G_2(i)$ is Δ. More formally, $(i+\Delta)' - i' = \Delta$, hence $(i+\Delta)' = i' + \Delta$. Then, for each i, if $G'(i) = \bigcup_{j=i}^{k} G_1(j)$, then $G'(i+\Delta) = \bigcup_{j=i+\Delta}^{i'+\Delta-1} G_1(j) = \bigcup_{j=i}^{i'-1} G_1(j+\Delta)$. By this result and considering $G_1 \trianglelefteq G'$, we conclude $G_1 \trianglelefteq G'$ with $P_{G'}^{G_1} = \Delta$. Note that by Proposition 9, $N_{G_1} = \frac{P_{G_1} \cdot N_{G_2}}{P_{G_2}}$, hence $P_{G'}^{G_1}$ is a multiple of Δ. Then, by Theorem 1, we have the thesis.

A.6.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \begin{cases} \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{if } l_{G_2} = l_{G_1}, \\ \{l'_{G_2}\} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}, & \text{otherwise,} \end{cases}$$

where l'_{G_2} is the greatest among the labels of $\mathcal{L}_{G_2}$ that are smaller than l_{G_2}. We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\hat{\mathcal{L}}_{G'} \subseteq \widetilde{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Then, since $k \in \widetilde{\mathcal{L}}_{G'}$, then $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ or $k = l'_{G_2}$.

If $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$, then, by definition of $\hat{\mathcal{L}}_{G_2}^{P_{G'}}$, $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_2} = k$. By definition of *Anchored-group*, $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ where k' is the first label of G_2 after k. Therefore $G'(k) \supseteq G_1(k)$. Since G_2 is a labeled aligned subgranularity of G_1 and since $k \in \mathcal{L}_{G_2}$, then $k \in \mathcal{L}_{G_1}$ and $G_1(k) = G_2(k)$. Hence $G'(k) \supseteq G_2(k)$. It follows that $\lceil h \rceil^{G'} = k$ and therefore, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

If $k = l'_{G_2}$, then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $l_{G_2} \neq l_{G_1}$. Therefore, since G_2 is a labeled aligned subgranularity of G_1 $l'_{G_2} < l_{G_1} < l_{G_2}$; then $\exists h$ with $0 < h < min(\lfloor l_{G_2} \rfloor^{G_2})$ s.t. $\lceil h \rceil^{G_1} = l_{G_1}$. Since, by definition of *Anchored-group*, $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$ and since $l'_{G_2} < l_{G_1} < l_{G_2}$, then $G'(l'_{G_2}) \supseteq G_1(l_{G_1})$. Hence $\lceil h \rceil^{G'} = l'_{G_2}$ and therefore, by definition of $\hat{\mathcal{L}}$, $l'_{G_2} = k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. If $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $k \in \widetilde{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

If $k \notin \hat{\mathcal{L}}_{G_2}^{P_{G'}}$, since $\nexists q \in \mathcal{L}_{G_2} \setminus \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ s.t. $min(\hat{\mathcal{L}}_{G_2}^{P_{G'}}) \leq q \leq max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$, then $k > max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ or $k < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$.

If $k > max(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ then, by definition of $\hat{\mathcal{L}}$, $min(\lfloor k \rfloor^{G_2}) > P_{G'}$. Since G_2 is a labeled aligned subgranularity of G_1 then $G_2(k) = G_1(k)$ and hence $min(\lfloor k \rfloor^{G_1}) > P_{G'}$. Since $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ then $min(\lfloor k \rfloor^{G'}) > P_{G'}$ in contrast with the hypothesis $k \in \hat{\mathcal{L}}_{G'}$.

If $k < min(\hat{\mathcal{L}}_{G_2}^{P_{G'}})$ then, by definition of l'_{G_2}, $k < l'_{G_2}$ or $k = l'_{G_2}$.

If $k < l'_{G_2}$ then, let k' be the next label of G_2 after k. Since $k < l'_{G_2}$ then, by definition l'_{G_2}, $k' \leq l'_{G_2}$. By definition of l'_{G_2} then $max(\lfloor l'_{G_2} \rfloor^{G_2}) \leq 0$. Since G_2 is a labeled aligned subgranularity of G_1 then $G_1(l'_{G_2}) = G_2(l'_{G_2})$; therefore $max(\lfloor l'_{G_2} \rfloor^{G_1}) \leq 0$. Since $G'(k) = \bigcup_{j=k}^{k'-1} G_1(j)$ and $k' \leq l'_{G_2}$, follows that $max(\lfloor k \rfloor^{G'}) \leq 0$ in contrast with the hypothesis $k \in \hat{\mathcal{L}}_{G'}$.

Finally if $k = l'_{G_2}$ then $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$. Since $k = l'_{G_2} \in \hat{\mathcal{L}}_{G'}$ then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = l'_{G_2}$. Since G' is the composition of granules of G_1, $\lceil h \rceil^{G_1}$ is defined. Let $q = \lceil h \rceil^{G_1}$. By definition of $\hat{\mathcal{L}}$, $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ and therefore $q \geq l_{G_1}$. Since, by definition of *Anchored-group*, G' is the composition of granules of G_1 and since $\lceil h \rceil^{G'} = l'_{G_2}$ and $\lceil h \rceil^{G_1} = q$, then $G_1(q) \subseteq G'(l'_{G_2})$. Therefore since $G'(l'_{G_2}) = \bigcup_{j=l'_{G_2}}^{l_{G_2}-1} G_1(j)$ then $q < l_{G_2}$. It follows that $l_{G_1} \leq q < l_{G_2}$ and hence $l_{G_1} \neq l_{G_2}$. By definition of $\widetilde{\mathcal{L}}_{G'}$, $l'_{G_2} = k \in \widetilde{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

A.7 Selecting operations

The selecting operations have a common part in the proof for the computation of the period length and the period label distance.

Let be $\Gamma = \frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$. The proof is divided into two steps: first we show that for each select operation if $i \in \mathcal{L}_{G'}$ then $i + \Gamma \in \mathcal{L}_{G'}$ (details for *Select-down*, *Select-up* and *Select-by-intersect* operations can be found below). The second step is the application of Theorem 1. Indeed, for each *Select* operation, the following holds: $\forall i \in \mathcal{L}_{G'}$ $G'(i) = G_1(i)$; this implies $G_1 \trianglelefteq G'$. From step 1 follows that $i + \Gamma \in \mathcal{L}_{G'}$, hence $G'(i + \Gamma) = G_1(i + \Gamma)$. By this result and considering $G_1 \trianglelefteq G'$, we conclude that $G_1 \,\bar{\trianglelefteq}\, G'$ with $P_{G'}^{G_1} = \Gamma$ which is a multiple of N_{G_1} by definition. Then, by Theorem 1 we have the thesis.

A.8 Proof of Proposition 6

A.8.1 Part 1

See Section A.7.

We prove that if $\lambda \in \mathcal{L}_{G'}$ then $\lambda' = \lambda + \Gamma \in \mathcal{L}_{G'}$.

By definition of the `select-down` operation, if $\lambda \in \mathcal{L}_{G'}$ then $\exists i \in \mathcal{L}_{G_2}$ s.t. $\lambda \in \Delta_k^l(S(i))$ where $S(i)$ is an ordered set defined as follows: $S(i) = \{j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i)\}$. In order to prove the thesis we need to show that $\exists i' \in \mathcal{L}_{G_2} | \lambda' \in \Delta_k^l(S(i'))$. Consider $i' = i + \frac{lcm(P_{G_1} P_{G_2}) N_{G_2}}{P_{G_2}}$ we will note that $i' \in \mathcal{L}_{G_2}$ (this is trivially derived from the periodicity of G_2). To prove that $\lambda' \in \Delta_k^l(S(i'))$ we show that $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements.

Indeed note that from periodicity of G_1, $\forall j \in S(i)$ if:

$$G_1(j) = \bigcup_{r=0}^{\tau_j} \perp(j_r) \tag{16}$$

then:

$$G_1\left(j'\right) = \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1} P_{G_2})) \tag{17}$$

Since $j \in S(i)$, $G_1(j) \subseteq G_2(i)$ then, from (16), $G_2(i) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r)$. Moreover, from periodicity of G_2:

$$G_2\left(i'\right) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1} P_{G_2})) \tag{18}$$

Since (17) and (18), $G_2\left(i'\right) \supseteq G_1\left(j'\right)$; hence $\forall j \in S(i), j' = (j + \Gamma) \in S(i')$. Analogously we can prove that $\forall j' \in S(i'), j = (j' - \Gamma) \in S(i)$.

Thus $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements; therefore if $j \in S(i)$ has position n in $S(i)$, so $j' \in S(i')$ has position n in $S(i')$. Hence it is trivial that if λ has position between k and $k+l-1$ in $S(i)$, then λ' has position between k and $k+l-1$ in $S(i')$. Hence if $\lambda \in \mathcal{L}_{G'}$, then $\lambda' \in \mathcal{L}_{G'}$.

A.8.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \bigcup_{i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}} \left\{ a \in A(i) | a \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} \right\};$$

where $\forall i \in \mathcal{L}_{G_2}$:

$$A(i) = \Delta_k^l \left(\{ j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i) \} \right).$$

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1)Suppose by contradiction that $\exists q \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. By definition of $\widetilde{\mathcal{L}}_{G'}$, $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$; therefore $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G_1} = q$. Moreover, by definition of $\widetilde{\mathcal{L}}_{G'}$ and by definition of *Select-down*, $\widetilde{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$ hence $q \in \mathcal{L}_{G'}$. Since, by definition of *Select-down* $G'(q) = G_1(q)$, then $\lceil h \rceil^{G'} = q$; hence, by definition of $\hat{\mathcal{L}}$, $q \in \hat{\mathcal{L}}_{G'}$ in contradiction with hypothesis.

(2)Suppose by contradiction that $\exists q \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $q \in \hat{\mathcal{L}}_{G'}$ then, by definition of *Select-down*

$$\exists i \in \mathcal{L}_{G_2} \, s.t. \, q \in \Delta_k^l \left(\{ j \in \mathcal{L}_{G_1} | \emptyset \neq G_1(j) \subseteq G_2(i) \} \right)$$

therefore, by definition of $A(i)$, $q \in A(i)$.

Since $q \in \hat{\mathcal{L}}_{G'}$ then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = q$. By definition of *Select-down*, $G'(q) = G_1(q)$, then $\lceil h \rceil^{G_1} = q$ and therefore $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$. Moreover, since $G_1(q) \subseteq G_2(i)$, then $\lceil h \rceil^{G_2} = i$ and therefore $i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$. Since $q \in A(i)$, $q \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ and $i \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$ then, by definition of $\widetilde{\mathcal{L}}_{G'}$, $q \in \widetilde{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

A.9 Proof of Proposition 7

A.9.1 Part 1

See Section A.7. We prove that if $i \in \mathcal{L}_{G'}$ then $i+\Gamma \in \mathcal{L}_{G'}$. From the periodicity of G_1, $i + \Gamma \in \mathcal{L}_{G_1}$ (this is trivially derived from the periodicity of G_1). Hence we only need to show that $\exists j' \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i + \Gamma)$. Since $i \in \mathcal{L}_{G'}$ then $\exists j \in \mathcal{L}_{G_2} | \emptyset \neq G_2(j) \subseteq G_1(i)$.

From the periodicity of G_2, if:

$$G_2(j) = \bigcup_{r=0}^{\tau_j} \perp(j_r) \tag{19}$$

then:

$$G_2\left(j + \frac{lcm(P_{G_1}P_{G_2})N_{G_2}}{P_{G_2}}\right) = \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1}P_{G_2})) \tag{20}$$

Moreover, from the (19) and since $G_1(i) \supseteq G_2(j)$:

$$G_1(i) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r)$$

From the periodicity of G_1:

$$G_1(i + \Gamma) \supseteq \bigcup_{r=0}^{\tau_j} \perp(j_r + lcm(P_{G_1}P_{G_2})) \tag{21}$$

From (20) and (21) follows that $G_1(i+\Gamma) \supseteq G_2\left(j + \frac{lcm(P_{G_1}P_{G_2})N_{G_2}}{P_{G_2}}\right)$, that is the thesis.

A.9.2 Part 2

Let

$$\widetilde{\mathcal{L}}_{G'} = \{i \in \hat{\mathcal{L}}_{G_1}^{P_{G'}} | \exists j \in \mathcal{L}_{G_2}\, s.t. \emptyset \neq G_2(j) \subseteq G_1(i)\};$$

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G_2}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$, then $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$; therefore $\exists h$ with $0 < h \leq P_{G'}$ s. t. $\lceil h \rceil^{G_1} = k$. Moreover, by

definition of $\widetilde{\mathcal{L}}_{G'}$ and by definition of *Select-down*, $\widetilde{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$ hence $q \in \mathcal{L}_{G'}$. Since, by definition of *Select-up*, $G'(k) = G_1(k)$, then $\lceil h \rceil^{G'} = k$. Hence, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$, in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$, then $\exists h$ with $0 < h \leq P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Since, by definition of *Select-up*, $G'(k) = G_1(k)$, then $\lceil h \rceil^{G_1} = k$; Therefore, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}^{P_{G'}}_{G_1}$. Moreover, since $k \in \hat{\mathcal{L}}_{G'}$ and $\hat{\mathcal{L}}_{G'} \subseteq \mathcal{L}_{G'}$, by definition of the *Select-up* operation, then $\exists j \in \mathcal{L}_{G_2}$ s.t. $\emptyset \neq G_2(j) \subseteq G_1(k)$. Hence by definition of $\widetilde{\mathcal{L}}_{G'}$, $k \in \widetilde{\mathcal{L}}_{G'}$, in contradiction with hypothesis.

A.10 Proof of Proposition 8

A.10.1 Part 1

See Section A.7. We prove that if $\lambda \in \mathcal{L}_{G'}$, then $\lambda' = \lambda + \Gamma \in \mathcal{L}_{G'}$.

By definition of the *select-by-intersect* operation, if $\lambda \in \mathcal{L}_{G'}$, then $\exists i \in \mathcal{L}_{G_2} : \lambda \in \Delta^l_k(S(i))$ where $S(i)$ is an ordered set defined as follows: $S(i) = \{j \in \mathcal{L}_{G_1} | G_1(j) \cap G_2(i) \neq \emptyset\}$. In order to prove the thesis we need to show that $\exists i' \in \mathcal{L}_{G_2} : \lambda' \in \Delta^l_k(S(i'))$. Consider $i' = i + \frac{lcm(P_{G_1} P_{G_2}) N_{G_2}}{P_{G_2}}$ note that $i' \in \mathcal{L}_{G_2}$ (this is trivially derived from the periodicity of G_2). To prove that $\lambda' \in \Delta^l_k(S(i'))$ we show that $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements.

Indeed note that $\forall j$ if $j \in S(i)$, then $G_1(j) \cap G_2(i) \neq \emptyset$. Hence $\exists l \in \mathbb{Z} : \perp(l) \subseteq G_1(j)$ and $\perp(l) \subseteq G_2(i)$. From the periodicity of G_1, $G_1(j + \Gamma) \supseteq \perp(l + lcm(P_{G_1} P_{G_2}))$. From the periodicity of G_2, $G_2(i') \supseteq \perp(l + lcm(P_{G_1} P_{G_2}))$. So $G_1(j + \Gamma) \cap G_2(i') \neq \emptyset$, therefore $\forall j \in S(i), (j + \Gamma) \in S(i')$.

Analogously we can prove that $\forall j' \in S(i'), (j' - \Gamma) \in S(i)$. Hence $S(i')$ is obtained from $S(i)$ by adding Γ to each of its elements. Therefore, if $j \in S(i)$ has position n in $S(i)$, then $j + \Gamma \in S(i')$ has position n in $S(i')$; hence if j has position between k and $k + l - 1$ in $S(i)$, then also $j + \Gamma$ has position between k and $k + l - 1$ in $S(i')$ and so $j + \Gamma \in \mathcal{L}_{G'}$.

A.10.2 Part 2

The proof is analogous to the ones of Proposition 6.

A.11 Set Operations

A.11.1 Proof of Proposition 9

Given the periodical granularities H and G with G label aligned subgranularity of H, we prove that $\frac{N_G}{P_G} = \frac{N_H}{P_H}$. The thesis is proved by considering the common period length of H and G i.e. $P_c = lcm(P_G, P_H)$.

Let N'_G be the difference between the label of the i^{th} granule of one period of G and the label of the i^{th} granule of the next period, considering P_c as the period length of G. Analogously N'_H is defined.

By periodicity of G, if $G(i) = \bigcup_{r=0}^{k} \perp(i_r)$ then $G(i+N'_G) = \bigcup_{r=0}^{k} \perp(i_r + P_c)$; since G is an aligned subranularity of H, $\forall i \in \mathcal{L}_H$ $H(i) = G(i) = \bigcup_{r=0}^{k} \perp(i_j)$ and, since H is periodic, $H(i + N'_H) = \bigcup_{r=0}^{k} \perp(i_j + P_c)$; from which we can easily derive that $i + N'_G = i + N'_H$, hence $N'_G = N'_H$.

From the definition of P_c, $\exists \alpha, \beta \in \mathbb{N}$ s. t. $\alpha P_H = \beta P_G$. Moreover, since $N'_H = N'_G$, then $\alpha N_H = \beta N_G$. Therefore $\frac{P_H}{N_H} = \frac{P_G}{N_G}$.

A.11.2 Property used in the proofs for set operations

Let Γ_1 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$ and Γ_2 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$. Since G_1 and G_2 are aligned subgranularity of a certain granularity H, from Proposition 9 we can easily derive that $\Gamma_1 = \Gamma_2$.

A.12 Proof of Proposition 10

A.12.1 Part 1

Union. Let Γ_1 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_1}}{P_{G_1}}$ and Γ_2 be $\frac{lcm(P_{G_1}, P_{G_2}) N_{G_2}}{P_{G_2}}$. The thesis will be proved by showing that $\forall i \in \mathcal{L}_{G'}$ if, $G'(i) = \bigcup_{r=0}^{k} \perp(i_r)$, then $G'(i + \Delta) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$ with $\Delta = \Gamma_1 = \Gamma_2$. Since $\mathcal{L}_{G'} = \mathcal{L}_{G_1} \cup \mathcal{L}_{G_2}$, two cases will be considered:

- $\forall i \in \mathcal{L}_{G_1}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_1, $G_1(i+\Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence $G'(i+\Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$.

- $\forall i \in \mathcal{L}_{G_2} - \mathcal{L}_{G_1}$ $G'(i) = G_2(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_2, $G_2(i+\Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence $G'(i+\Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$.

Since $\Gamma_1 = \Gamma_2$, then $\forall i \in \mathcal{L}_{G'}$ if $G'(i) = \bigcup_{r=0}^{k} \perp(i_r)$, then $G'(i + \Gamma_1) = G'(i + \Gamma_2) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$. Hence, by definition of $\trianglelefteq$, we have the thesis.

Intersect. $\forall i \in \mathcal{L}_{G'} = \mathcal{L}_{G_1} \cap \mathcal{L}_{G_2}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. From the periodicity of G_1 and G_2, $i + \Gamma_1 \in \mathcal{L}_{G_1}$ e $i + \Gamma_2 \in \mathcal{L}_{G_2}$; since $\Gamma_1 = \Gamma_2$, then $i + \Gamma_1 \in \mathcal{L}_{G'}$. Moreover $G'(i+\Gamma_1) = G_1(i+\Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence, by the definition of $\trianglelefteq$, we have the thesis.

Difference. $\forall i \in \mathcal{L}_{G'} = \mathcal{L}_{G_1} - \mathcal{L}_{G_2}$ $G'(i) = G_1(i) = \bigcup_{r=0}^{k} \perp(i_r)$. Since $i \in \mathcal{L}_{G_1}$ from the periodicity of G_1 $i + \Gamma_1 \in \mathcal{L}_{G_1}$. Since $i \notin \mathcal{L}_{G_2}$, from the periodicity of G_2, $i + \Gamma_2 \notin \mathcal{L}_{G_2}$ (if it would exists $i + \Gamma_2 \in \mathcal{L}_{G_2}$, from periodicity of G_2 would exists $i \in \mathcal{L}_{G_2}$ that is not possible for hypothesis). Hence $i + \Gamma_1 \in \mathcal{L}_{G'}$. Moreover $G'(i+\Gamma_1) = G_1(i+\Gamma_1) = \bigcup_{r=0}^{k} \perp(i_r + lcm(P_{G_1}, P_{G_2}))$; hence, by the definition of $\trianglelefteq$, we have the thesis.

A.12.2 Part 2

Let $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G_1}^{P_{G'}} \cup \hat{\mathcal{L}}_{G_2}^{P_{G'}}$.

We show that $\widetilde{\mathcal{L}}_{G'} = \hat{\mathcal{L}}_{G'}$ by proving that (1) $\widetilde{\mathcal{L}}_{G'} \subseteq \hat{\mathcal{L}}_{G'}$ and (2) $\widetilde{\mathcal{L}}_{G'} \supseteq \hat{\mathcal{L}}_{G'}$.

(1) Suppose by contradiction that $\exists k \in \widetilde{\mathcal{L}}_{G'} \setminus \hat{\mathcal{L}}_{G'}$. Since $k \in \widetilde{\mathcal{L}}_{G'}$ then $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ or $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$. Suppose that $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ (the proof is analogous if $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$). Since $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$, then $\exists\ 0 < h < P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Since, by definition of the *Union* operation $G'(k) = G_1(k)$, then $\lceil h \rceil^{G'} = k$. Hence, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G'}$ in contrast with the hypothesis.

(2) Suppose by contradiction that $\exists k \in \hat{\mathcal{L}}_{G'} \setminus \widetilde{\mathcal{L}}_{G'}$. Since $k \in \hat{\mathcal{L}}_{G'}$, then, by definition of $\hat{\mathcal{L}}$, $\exists\ 0 < h < P_{G'}$ s.t. $\lceil h \rceil^{G'} = k$. Moreover, by definition of the *Union* operation, $k \in \mathcal{L}_{G_1}$ or $k \in \mathcal{L}_{G_2}$. Suppose that $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$ (the proof is analogous if $k \in \hat{\mathcal{L}}_{G_2}^{P_{G'}}$). By definition of the *Union* operation, $G'(k) = G_1(k)$ therefore $\lceil h \rceil^{G_1} = k$ and so, by definition of $\hat{\mathcal{L}}$, $k \in \hat{\mathcal{L}}_{G_1}^{P_{G'}}$. Hence, by definition of $\widetilde{\mathcal{L}}$, $k \in \widetilde{\mathcal{L}}_{G'}$ in contradiction with the hypothesis.

Proc. of the 12th International Symposium on Temporal Representation and Reasoning (TIME)

Recent Advances in Constraints, Revised selected papers from the Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)

Lecture Notes in Computer Science

Proc. of the 11th International Symposium on Temporal Representation and Reasoning (TIME)

Proc. of the 9th European Conference on Logics in Ar- ti- fi-